# COBOL

# Micro News

## CURSO DE COMPUTAÇÃO

O ELO ENTRE O HOMEM E A MÁQUINA

Fabrício Rocha de Sousa

**Coordenador Responsável:** Antonio Eduardo Martinelli

**Equipamento:** Microcomputadores PC (Turbo) IBM compatíveis - 640 kb de memória

**Cursos ministrados:** BASIC, COBOL, dBASE III, LOTUS 1-2-3 e WORDSTAR.

# COBOL PARA MICROS PC

ANTONIO EDUARDO MARTINELLI

ANTONIO JOSÉ DOS SANTOS

*"Se não houver frutos,*
*valeu a beleza das flores.*

*Se não houver flores,*
*valeu a sombra das folhas.*

*Se não houver folhas,*
*valeu a intenção da semente".*

*A semente está lançada, agora cabe a*
*você, leitor, fazer com que ela*
*dê origem a firmes ramos que*
*sustentam os frutos de seu aprendizado.*

Antonio Eduardo Martinelli

# ESTRUTURA FUNCIONAL BÁSICA DE UM ARQUIVO

— ARQUIVO
— REGISTRO
— CAMPO

## 1) Estrutura Funcional Básica de um Arquivo

### 1.1) Arquivo

Um arquivo consiste de um conjunto de registros agrupados segundo uma organização lógica.

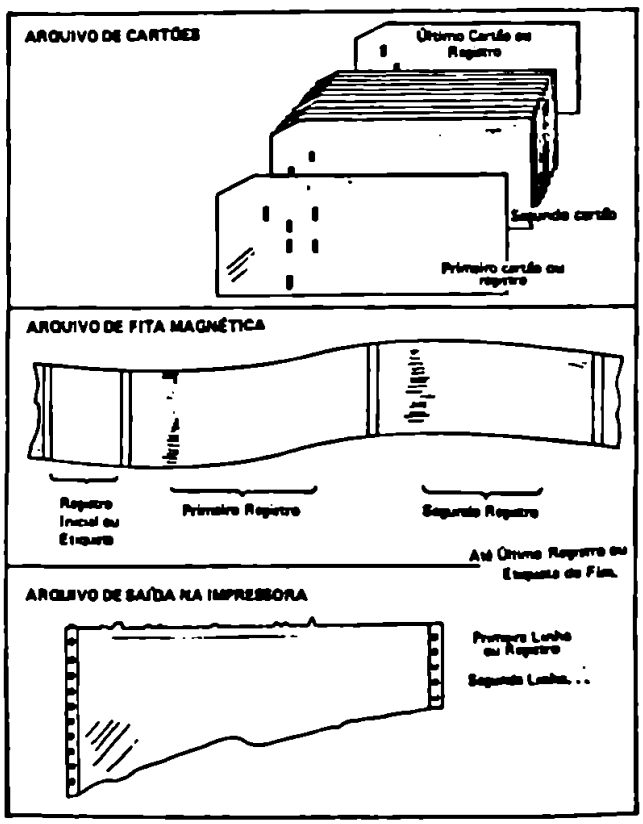
*Exemplo:* Conjunto de fichas de registro de funcionários de uma empresa.

Os arquivos podem ser organizados de diversas formas, dependendo do dispositivo utilizado.

| | |
|---|---|
| FITAS MAGNÉTICAS<br>ARQUIVOS DE CARTÕES PERFURADOS | ARQUIVOS SEQUENCIAS |

| | |
|---|---|
| DISCOS MAGNÉTICOS | ARQUIVOS SEQUENCIAIS<br>ARQUIVOS DE ACESSO DIRETO<br>ARQUIVOS SEQUENCIAIS-INDEXADOS |

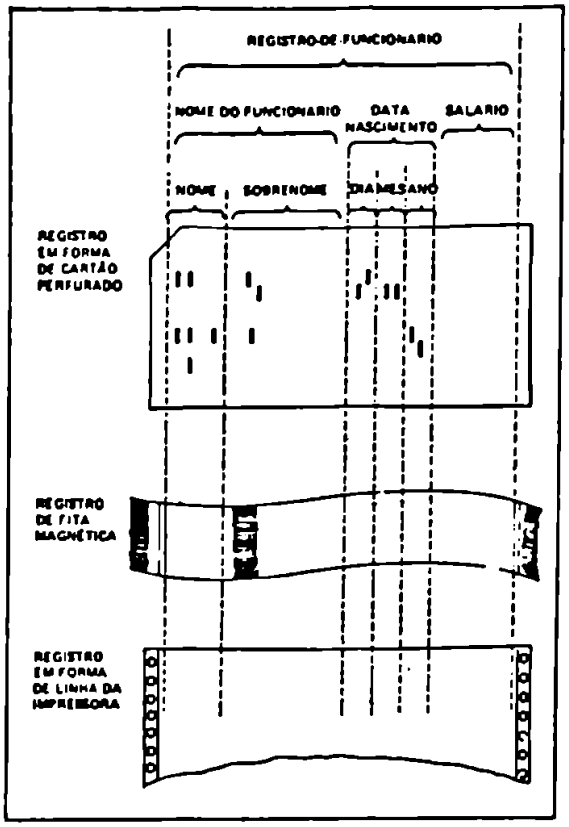
- Arquivo de Entrada – Utilizado como entrada de dados em um programa.
- Arquivo de Saída – Constituido de resultados obtidos após a ação de um programa. Pode aparecer em forma de cartão-perfurado, gravado em disco ou fita magnética e também em formato de relatório impresso.



## 1.2) Registro

Um registro é formado por diversos campos referentes a um mesmo assunto. É identificado através de um nome, chamado NOME-DE-REGISTRO e seu tamanho é dado pela soma dos comprimentos de seus campos formadores.

*Exemplo:* Cada ficha de registro de funcionário.



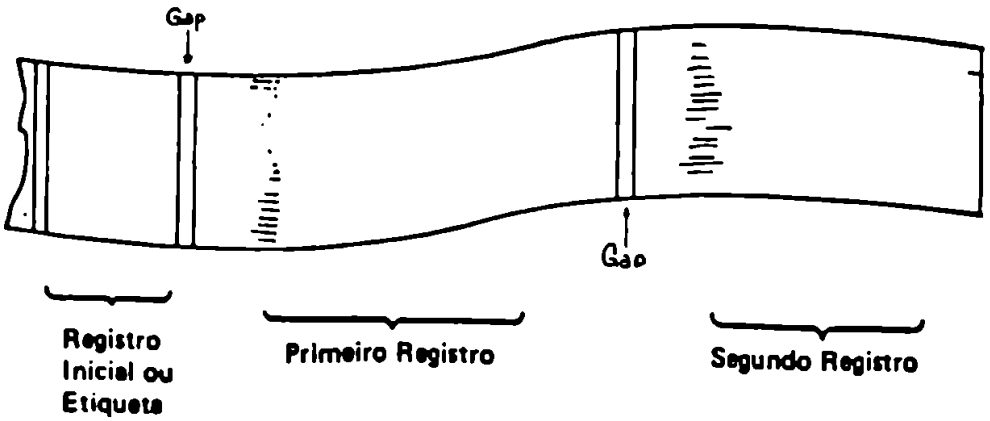

Nos arquivos podemos encontrar dois tipos de registros. A saber:

- Registro físico.
- Registro lógico.

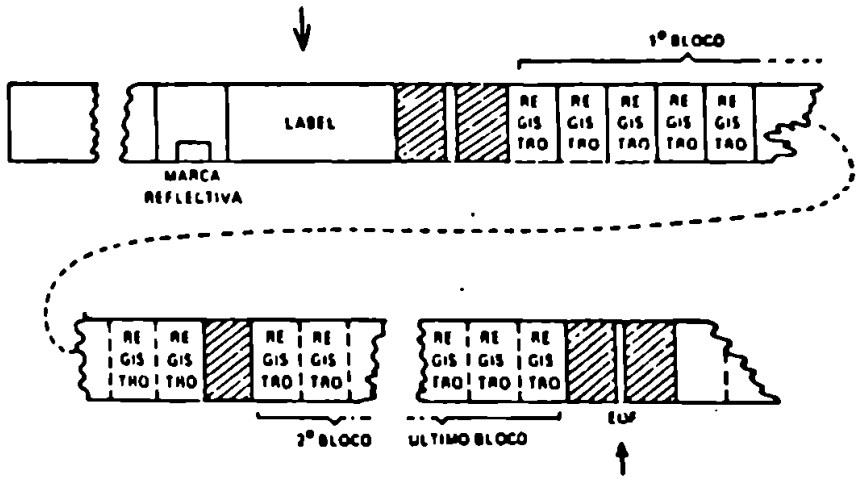
O registro lógico é aquele que contém a descrição de seus campos constituintes.

O registro físico, chamado de bloco, pode conter um ou mais registros lógicos dependendo das características de armazenamento utilizadas. Entre dois blocos existe um intervalo chamado GAP.

ARQUIVO DE FITA MAGNÉTICA



O primeiro e o último registros de um arquivo possuem um formato diferente dos demais e servem para identificar os pontos terminais (início e fim) desse arquivo. São chamados de **Etiquetas** ou **Labels**.




### 1.3) Campo

É cada uma das partes integrantes de um registro.

*Exemplo:* Na ficha de registro de um funcionário, poderíamos ter os seguintes campos:

- número de funcionário.
- nome do funcionário.
- endereço.
- data de nascimento.
- salário inicial.

Um campo possui obrigatoriamente as seguintes características:

- Nome: nome através do qual o campo é reconhecido.
- Tipo: Numérico – Contém somente algarismos, " + ", " - ", " . " ou " , ".

*Exemplo:* Salário Inicial.

Alfabético – Contém somente letras de alfabeto e o espaço em branco, identificado aqui por "ϕ". *Exemplo:* nome do funcionário.

Alfanumérico – Contém letras, algarismos e caracteres especiais. *Exemplo:* endereço.

- Comprimento: tamanho designado ao campo. Medido em "número de caracteres.
- Valor: é o conteúdo do campo.

*Exemplo:* CEP Ø 3 Ø 8 Ø

Nome – CEP

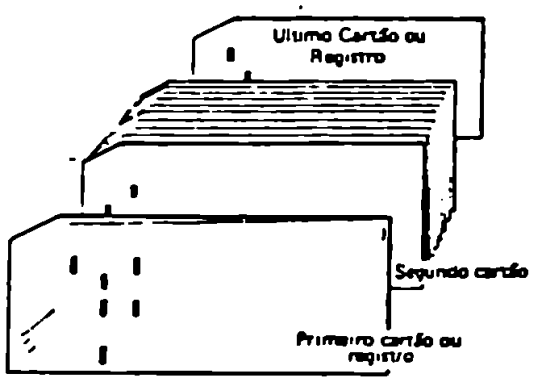
Tipo – numérico

Comprimento – 5 caracteres.

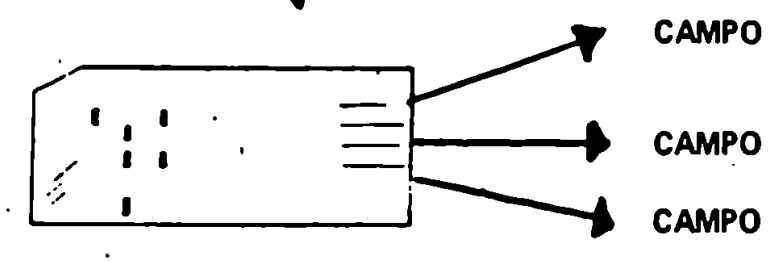
Valor – Ø 3 Ø 8 Ø

OBS. Devido a diferença de representação binária entre caracteres numéricos e não-numéricos, um campo numérico de mesmo tamanho que um campo alfabético ou alfanumérico ocupa menos espaço na memória de um computador.

**ARQUIVO DE CARTÕES**




**REGISTRO**



# CAPÍTULO 2

# INTRODUÇÃO AO COBOL

— A LINGUAGEM COBOL
— AS DIVISÕES DO COBOL E SUAS RESPECTIVAS SEÇÕES
— CONJUNTO DE CARACTERES/REGRAS DE PONTUAÇÃO
— TIPOS DE PALAVRAS
— FOLHA DE CODIFICAÇÃO

## 2) Introdução ao Cobol

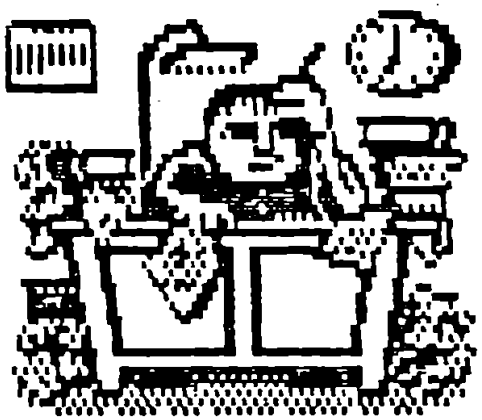
### 2.1) A Linguagem COBOL

COBOL (COMMON BUSINESS ORIENTED LANGUAGE – Linguagem Orientada para Aplicações Comerciais) foi inicialmente desenvolvida no final da década de 50 com o objetivo de padronizar os métodos de programação comercial.

Após sua primeira versão surgida em agosto de 1961, o COBOL tem sido sucessivamente melhorado e novas versões cada vez mais potentes têm surgido.

O COBOL, como já dissemos, é uma linguagem voltada especificamente para fins comerciais. É autodescritiva e com estruturação bem definida, desenvolvida a fim de facilitar tanto a escrita como a leitura.

Possui uma divisão interna de tal forma que cada seção contribui com uma função específica dentro das divisões, distribuindo o fluxo de processamento entre elas.



### 2.2) As divisões de COBOL e suas respectivas seções.

Um programa COBOL possui internamente quatro divisões, cada qual por sua vez, subdividida em seções e estas em parágrafos.

O "esqueleto" de um programa COBOL seria o seguinte:

**IDENTIFICATION DIVISION**

Função: identificação do programa

Seções: não possui

**ENVIRONMENT DIVISION**

Função: descrição dos equipamentos e informações sobre os arquivos que serão utilizados.

Seções:
1 - **CONFIGURATION SECTION** – Características de sistema.
2 - **INPUT-OUTPUT SECTION** – identificação dos arquivos.

**DATA DIVISION**

Função: Descrição detalhada dos dados a serem utilizados (arquivos e áreas de trabalho).

Seções:
1 - **FILE SECTION** – especificação dos arquivos de entrada e saída (E/S).
2 - **WORKING-STORAGE SECTION** – especificação das áreas de trabalho e definição de constantes.
3 - **LINKAGE SECTION** – especificação dos parâmetros de sub-rotinas.
4 - **SCREEN SECTION** – definição de telas.

**PROCEDURE DIVISION**

Função: Contém o processamento de dados propriamente dito.

Seções: Podem ser criadas pelo programador em caráter não orbigatório.

OBS.:
1) As divisões do COBOL não podem aparecer fora da ordem descrita acima.
2) As seções da DATA DIVISION são opcionais, no entanto, quando aparecerem, devem respeitar a ordem acima.

2.3) **Conjunto de caracteres / Regras de pontuação.**

Os caracteres aceitos pela linguagem COBOL são:

- dígitos de Ø a 9
- letras do alfabeto de A a Z incluindo K, Y, W.
- símbolos aritméticos " + ", " ( ", " ) ", " - ", " *", " / ", " * *" (exponenciação) e " = ".
- pontuação: " . ", " , ", " ; "
- símbolos relacionais " <", " >"
- caracteres de edição (EX: $)
- hífen ( - )

**Regras de pontuação**

- Qualquer frase deve terminar por um ponto.
- As palavras podem ser delimitadas pelos seguintes caracteres: " ß ", " . ", " ; ", " , " ou parênteses.
- Os parênteses e os símbolos relacionais são isolados por "ß".
- Um ponto, uma vírgula ou um ponto e vírgula devem ser SEMPRE seguidos por um "ß".

2.4) **Tipos de palavras**

- **Palavras Reservadas** ... São palavras com significado especial para o COBOL. Dividem-se em:
  - **Palavras-Chaves:** devem ser escritas obrigatoriamente para a execução do programa.

*Exemplo:* Comandos – aparecem na **PROCEDURE DIVISION.**
Constantes figurativas – representam valores conhecidos.

| TABELA DE CONSTANTES FIGURATIVAS | |
|---|---|
| CONSTANTE | VALOR |
| ZERO(S) (ES) | Ø |
| SPACE(S) | ß |
| LOW-VALUE | MENOR VALOR NUMÉRICO ASSUMIDO PELO COBOL |
| HIGH-VALUE | MAIOR VALOR NUMÉRICO ASSUMIDO PELO COBOL |
| QUOTE | OCORRÊNCIA DE ASPAS |
| "LITERAL" | SEQUÊNCIA FORMADA PELO LITERAL INDICADO |

*Exemplo:* **QUOTE "BRASIL" QUOTE** (representa "BRASIL" com o uso das aspas).

- **ALL "A"** (representa um campo preenchido totalmente com A).
  - **Palavras opcionais** – não são necessárias para a execução do programa. Quando aparecem, servem apenas para melhorar o entendimento do mesmo.
- **Palavras do programador** – são nomes característicos fornecidos pelo próprio programador.
  - **Regras de Formação:**
    - Caracteres permitidos: letras (A-Z), dígitos (Ø-9), hífen ( - ).

Cuidado: O "ß" não é permitido!

— Tamanho máximo — 3Ø caracteres. Excessão: O COBOL só identifica os 6 primeiros caracteres do nome do programa.
— Não podem começar nem terminar por hífen.
— Deve começar por letra (exceto nomes de parágrafos ou seções).
— Não podem ser palavras reservadas.

*Exemplo:* Nome-de-Arquivo (ARQ-ENT).
Nome-de-Registro (REG-SAI).
Nome-de-Parágrafo (ØØ1-P).

- **Literais** — São os valores que as diversas variáveis podem assumir.
  Dividem-se em:
  - **Numéricas** — com máximo de 18 caracteres.
    *Exemplo:* 1ØØØ
    1Ø.1Ø
    — 3.1416
  - **Alfamunéricas** — com máximo de 12Ø caracteres, são delimitados por aspas ou apóstrofo.
    *Exemplo:* "MICRO NEWS"
    "COBOL"

**2.5) Folha de codificação.**

A folha de codificação COBOL possui 72 colunas úteis e é utilizada para que escrevamos o programa de forma a respeitarmos as exigências da linguagem quanto à colocação das partes integrantes das linhas desse programa.

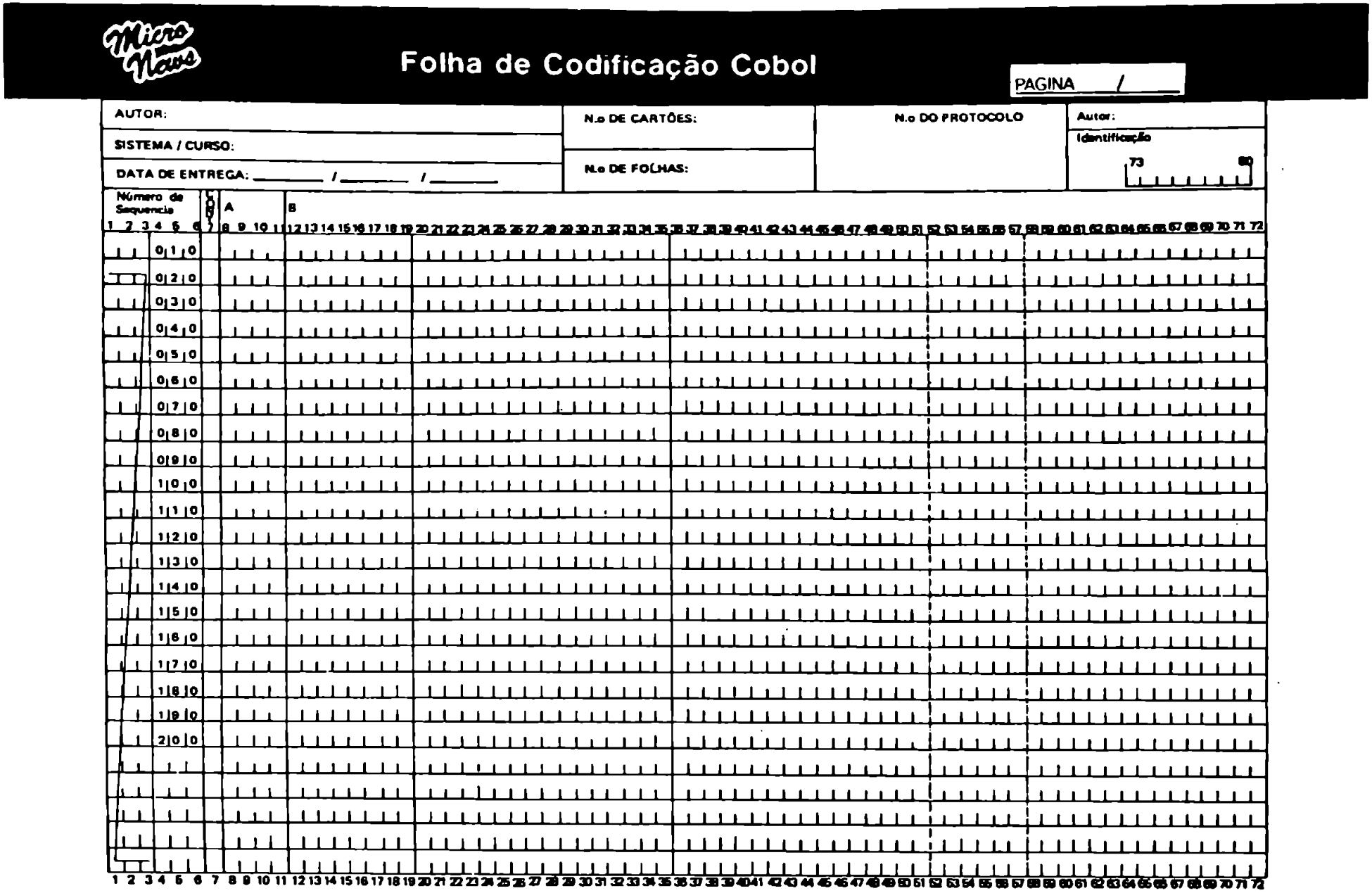
É dividida em quatro partes:

— Coluna de 1 a 6 — Utilizados para a numeração (opcional) em ordem crescente das linhas de um programa.
— Coluna 7 — pode conter apenas um dos três símbolos:
" - ": Continuação de linha anterior (no caso de literais não-numéricas).
" *": linha abandonada para comentários. Não é interpretada.
" / ":Indica mudança de página na listagem do programa-fonte.
— Colunas 8 a 72: Disponíveis para as entradas do programa. Por sua vez, dividem-se em:
- Margem A — colunas 8 a 11.
- Margem B — colunas 12 a 72.

A Margem A é utilizada para as seguintes entradas:

— Nomes de divisões.
— Nomes de seções.
— Nomes de parágrafos.
— descrições de arquivos e números-de-nível (calma, vocês verão o que é isso).

Todas as demais entradas serão feitas pela margem B (coluna 12).


Micro Naus
Folha de Codificação Cobol
PAGINA /
AUTOR:
SISTEMA / CURSO:
DATA DE ENTREGA: / /
N.o DE CARTÕES:
N.o DE FOLHAS:
N.o DO PROTOCOLO
Autor:
Identificação
73
80
Número de Sequencia
A
B

# CAPÍTULO 3

# IDENTIFICATION DIVISION

## 3) IDENTIFICATION DIVISION

É a primeira divisão do COBOL. Serve como identificação e traz informações sobre o programa.

Como vocês já sabem, ela não possui seções. Seus parágrafos são:

- **PROGRAM-ID:** é o único parágrafo obrigatório dessa divisão. Identifica o nome do programa que está sendo elaborado.
  Lembrem-se! Só são considerados os 6 primeiros caracteres desse nome.

  FORMATO: **PROGRAM-ID.** *␣nome do programa.*

- **AUTHOR:** segue-se o nome do programador responsável.

  FORMATO: **AUTHOR.** *␣nome do programador.*

- **INSTALLATION:** nome da empresa ou local de uso ou geração do programa.

  FORMATO: **INSTALLATION.** *␣local de instalação.*

- **DATE-WRITTEN:** data em que o programa foi escrito.

  FORMATO: **DATE-WRITTEN.** *␣data de escrita.*

- **DATE-COMPILED:** data em que o programa foi compilado. Algumas versões do COBOL adicionam a data de compilação se esta cláusula estiver presente.

  FORMATO: **DATE-COMPILED.** *␣data de compilação.*

- **SECURITY:** comentários sobre a segurança do programa e/ou seus arquivos.

  FORMATO: **SECURITY.** *␣comentários.*

- **REMARKS:** parágrafo destinado, também, a comentários.

  FORMATO: **REMARKS.** *␣comentários.*

*Com* exceção do parágrafo **PROGRAM-ID**, todos os demais são opcionais, encarados pelo COBOL como comentários. No entanto, se estiverem presentes, devem respeitar a ordem descrita acima.

# Folha de Codificação Cobol

PAGINA 01 / 01

| | | | |
|---|---|---|---|
| AUTOR: EDUARDO | N.o DE CARTÕES: | N.o DO PROTOCOLO | Autor: EDUARDO |
| SISTEMA / CURSO: EXEMPLO/MICRO-NEWS | N.o DE FOLHAS: 01 | | Identificação 73 80 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

| Número da Sequencia | | A | B |
|---|---|---|---|
| 1 2 3 4 5 6 | 7 | 8 9 10 11 | 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 |
| 010 | | IDENTIFICATION DIVISION. | |
| 020 | | PROGRAM-ID. EX1-01. | |
| 030 | | AUTHOR. EDUARDO. | |
| 040 | | INSTALLATION. MICRO NEWS. | |
| 050 | | DATE-WRITTEN. 12/06/88. | |
| 060 | | DATE-COMPILED. | |
| 070 | | SECURITY. SEM DISPOSITIVOS DE SEGURANCA. | |
| 080 | | REMARKS. EXEMPLO DA PRIMEIRA DIVISAO DE UM PROGRAMA. | |
| 090 | | | |
| 100 | | | |
| 110 | | | |
| 120 | | | |
| 130 | | | |
| 140 | | | |
| 150 | | | |
| 160 | | | |
| 170 | | | |
| 180 | | | |
| 190 | | | |
| 200 | | | |

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72

Espaço reservado para anotações de aula.

Espaço reservado para anotações de aula.

# CAPÍTULO 4

# ENVIRONMENT DIVISION

— CONFIGURATION SECTION
— IMPUT - OUTPUT SECTION

## 4) ENVIRONMENT DIVISION

É a segunda divisão do COBOL. Responsável pela descrição do computador utilizado, designação dos arquivos a seus periféricos correspondentes e também pela definição de técnicas de entrada e saída (E/S).

Possui 2 seções: **CONFIGURATION SECTION** e **INPUT-OUTPUT SECTION.**

### 4.1) CONFIGURATION SECTION

Fornece informações sobre o tipo de computador utilizado. Possui 3 parágrafos.

- **SOURCE-COMPUTER:** especifica qual o computador a ser utilizado na compilação do programa.

  FORMATO: **SOURCE-COMPUTER.** *ßcomputador-fonte.*

- **OBJECT-COMPUTER:** especifica qual o computador onde o programa será executado.

  FORMATO: **OBJECT-COMPUTER.** *ßcomputador-objeto.*

Este parágrafo possui ainda as cláusulas **MEMORY SIZE** e **SEGMENT-LIMIT** que não serão discutidas neste trabalho.

- **SPECIAL-NAMES:** estabelece uma ligação entre termos internos do COBOL e nomes externos (mnemônicos) atribuidos pelo programador.

  FORMATO GERAL:

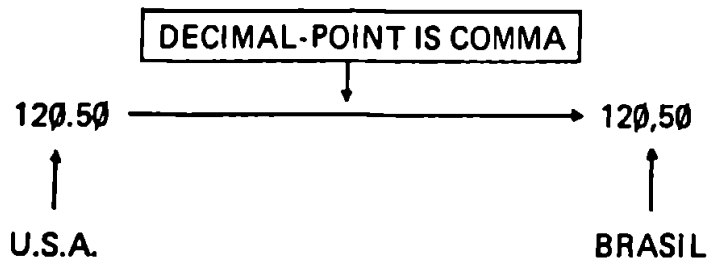
| A | B |
|---|---|
| SPECIAL | NAMES. |
| | CURRENCY SIGN IS *literal.*; |
| | DECIMAL-POINT IS COMMA. |
| | PRINTER IS *mnemônico.* |

Somente são especificados aqueles que serão utilizados pelo programa. Não é necessário que todos os três apareçam.

- **CURRENCY SIGN:** substitui na cláusula PICture (vejam Cap. 5) o sinal corrente "$" pela literal especificada.

- **DECIMAL-POINT IS COMMA:** substitui o ponto-decimal utilizado na notação americana, pela vírgula utilizada na notação nacional.

*Exemplo:*



- **PRINTER IS:** define o mnemônico utilizado na cláusula **DISPLAY UPON** (PROCEDURE DIVISION).

Tanto a **CONFIGURATION SECTION** como os seus parágrafos constituintes são opcionais.

Os parágrafos **SOURCE-COMPUTER** e **OBJECT-COMPUTER** são considerados comentários.

### 4.2) INPUT-OUTPUT SECTION

Define os arquivos a serem utilizados pelo programa, suas unidades de entrada ou saída e técnicas especiais de utilização. Possui 2 parágrafos:

- **FILE-CONTROL:** seleciona os arquivos utilizados e seus periféricos (unidades de entrada ou saída) correspondentes.

FORMATO:

```
A          |  B
           |
FILE-      |  CONTROL.
           |  SELECT arquivo-1 ASSIGN TO periférico-1
           |         FILE-STATUS IS chave-1
           |         ACCESS MODE IS modo de acesso
           |         ORGANIZATION IS organização
           |         RECORD KEY IS chave-2.
           |  SELECT arquivo-2 ASSIGN TO periférico-2
           |    .          .
           |    .          .
           |    .          .
```

Voltaremos a este assunto mais adiante.

- **I-O-CONTROL:** aplicada somente a arquivos de acesso direto. Especifica os arquivos utilizados em momentos diferentes da execução do programa, solicitando as mesmas áreas de memória para eles.

| *ORGANIZAÇÃO* | *ACESSO* | *CARACTERÍSTICAS* |
|---|---|---|
| SEQÜENCIAL | SEQÜENCIAL | 1. Os registros só podem ser pesquisados um a um. Para pesquisa de um registro, é necessário que se passe pelos anteriores. |
| INDEXADA | SEQÜENCIAL<br>RANDÔMICO | Idem 1<br>2. Os registros não precisam ser pesquisados um a um, não é necessário que se passe pelos anteriores. |
| DIRETA | SEQÜENCIAL<br>RANDÔMICO | Idem 1<br>Idem 2 |

Espaço reservado para anotações de aula.

# CAPÍTULO 5

# DATA DIVISION

— FILE SECTION
— WORKING - STORAGE SECTION

## 5) DATA DIVISION

É a terceira divisão do COBOL. Sua função é especificar cada dado, cada registro e cada arquivo que serão utilizados na **PROCEDURE DIVISION.**

Ela reserva na memória espaço para cada um desses ítens, atribuindo nome, tipo e tamanho às informações.

Enfim, é na **DATA DIVISION** que as informações são preparadas para o processamento do programa.

Possui 4 seções:

- **FILE SECTION:** descreve o conteúdo e a organização interna de todos os arquivos utilizados no programa.

- **WORKING-STORAGE SECTION:** descreve as áreas de trabalho que serão utilizadas, ou seja, aqueles dados que só existem durante a execução do programa (não pertencem a arquivos externos).

- **LINKAGE SECTION:** só aparece em programas utilizados como subrotinas e serve para descrever os parâmetros transferidos pelo programa principal.

- **SCREEN SECTION:** responsável pelas definições de telas utilizadas durante a execução do programa. Será discutida no próximo capítulo.

### 5.1) FILE SECTION

Como discutido acima, é nesta seção onde os arquivos a serem manipulados pelo programa são especificados.

Para cada arquivo atribuido a uma cláusula **SELECT** da **CONFIGURATION SECTION (ENVIRONMENT DIVISION)** deve existir uma entrada de descrição de arquivo.

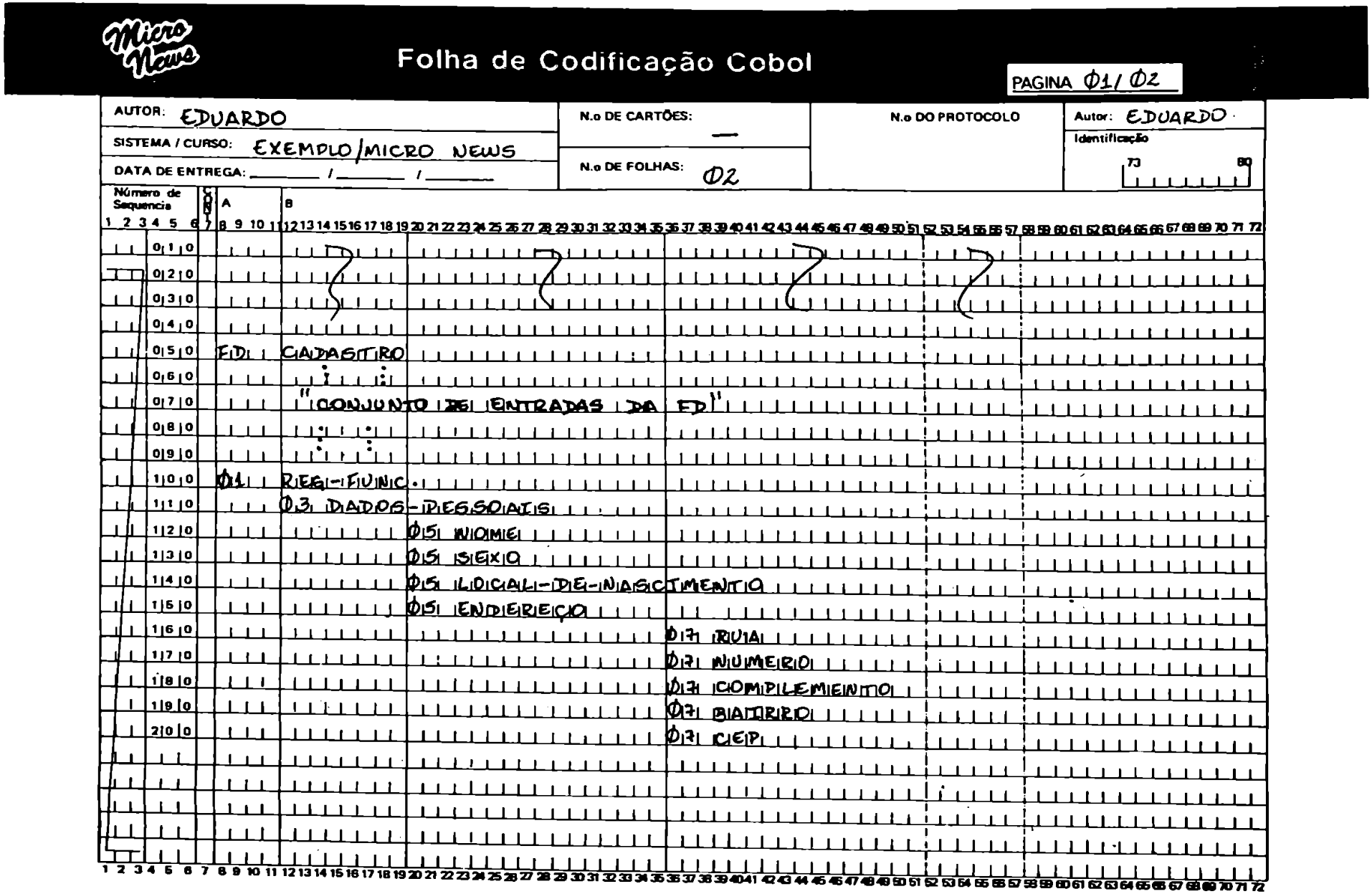
#### 5.1.1) Números e indicadores de nível.

O início de uma descrição de arquivo é especificado através de um indicador de nível do tipo FD ( **FILE DESCRIPTION** – descrição de arquivo) ou **SD** ( **SORT-FILE DESCRIPTION** – descrição de arquivo sort).

As subdivisões dos registros são estruturadas através dos números de nível. Um registro possui as seguintes subdivisões:

- ítens elementares: são aqueles que não possuem subdivisões.

- ítens de grupo: formados por uma determinada sequência de ítens ou grupos elementares.



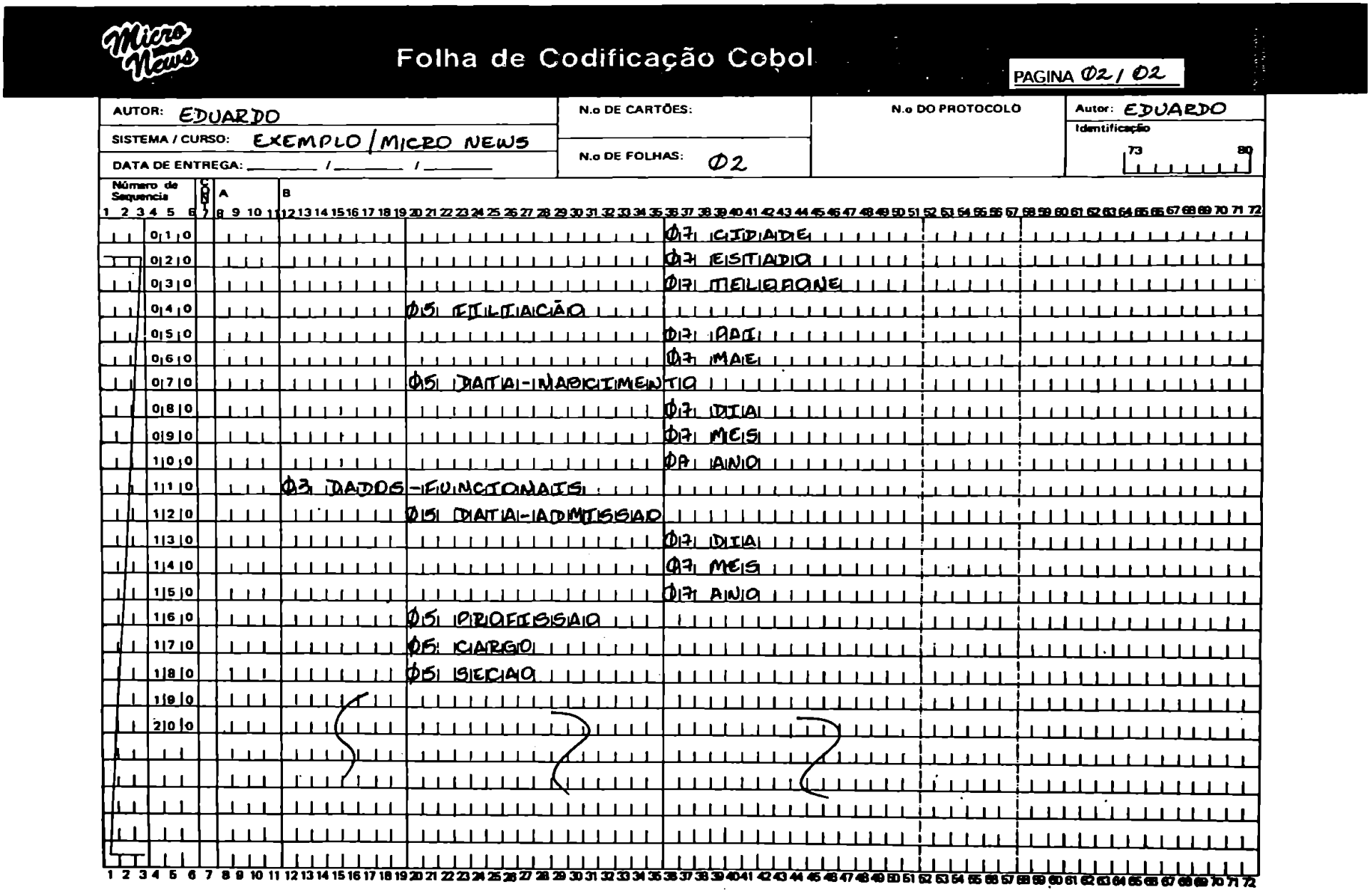
Micro News

# Folha de Codificação Cobol

PAGINA Ø1/ Ø2

| | | | |
|---|---|---|---|
| AUTOR: EDUARDO | N.o DE CARTÕES: — | N.o DO PROTOCOLO | Autor: EDUARDO |
| SISTEMA / CURSO: EXEMPLO/MICRO NEWS | N.o DE FOLHAS: Ø2 | | Identificação 73 80 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

| Número de Sequencia (1 2 3 4 5 6) | CONT (7) | A (8 9 10 11) | B (12 13 14 15 16 17 18 19) | (20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35) | (36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51) | (52 53 54 55 56 57) | (58 59 60 61 62 63 64 65 66 67 68 69 70 71 72) |
|---|---|---|---|---|---|---|---|
| 010 | | | | | | | |
| 020 | | | | | | | |
| 030 | | | | | | | |
| 040 | | | | | | | |
| 050 | | FD | CADASTRO | | | | |
| 060 | | | | | | | |
| 070 | | | "CONJUNTO DE ENTRADAS DA FD" | | | | |
| 080 | | | | | | | |
| 090 | | | | | | | |
| 100 | | Ø1 | REG-FUNC. | | | | |
| 110 | | | Ø3 DADOS-PESSOAIS | | | | |
| 120 | | | | Ø5 NOME | | | |
| 130 | | | | Ø5 SEXO | | | |
| 140 | | | | Ø5 LOCAL-DE-NASCIMENTO | | | |
| 150 | | | | Ø5 ENDEREÇO | | | |
| 160 | | | | | Ø7 RUA | | |
| 170 | | | | | Ø7 NUMERO | | |
| 180 | | | | | Ø7 COMPLEMENTO | | |
| 190 | | | | | Ø7 BAIRRO | | |
| 200 | | | | | Ø7 CEP | | |



Micro News

# Folha de Codificação Cobol

PAGINA 02 / 02

| | | | |
|---|---|---|---|
| AUTOR: EDUARDO | N.o DE CARTÕES: | N.o DO PROTOCOLO | Autor: EDUARDO |
| SISTEMA / CURSO: EXEMPLO / MICRO NEWS | N.o DE FOLHAS: 02 | | Identificação 73 80 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

| Número de Sequencia (1 2 3 4 5 6) | CONT (7) | A (8 9 10 11) | B (12 13 14 15 16 17 18 19) | (20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35) | (36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51) | (52 53 54 55 56 57) | (58 59 60 61 62 63 64 65 66 67 68 69 70 71 72) |
|---|---|---|---|---|---|---|---|
| 010 | | | | | 07 CIDADE | | |
| 020 | | | | | 07 ESTADO | | |
| 030 | | | | | 07 TELEFONE | | |
| 040 | | | | 05 FILIACÃO | | | |
| 050 | | | | | 07 PAI | | |
| 060 | | | | | 07 MAE | | |
| 070 | | | | 05 DATA-NASCIMENTO | | | |
| 080 | | | | | 07 DIA | | |
| 090 | | | | | 07 MES | | |
| 100 | | | | | 07 ANO | | |
| 110 | | | 03 DADOS-FUNCIONAIS | | | | |
| 120 | | | | 05 DATA-ADMISSAO | | | |
| 130 | | | | | 07 DIA | | |
| 140 | | | | | 07 MES | | |
| 150 | | | | | 07 ANO | | |
| 160 | | | | 05 PROFISSAO | | | |
| 170 | | | | 05 CARGO | | | |
| 180 | | | | 05 SECAO | | | |
| 190 | | | | | | | |
| 200 | | | | | | | |

No exemplo acima, temos:

- Arquivo: cadastro. Identificado pela **FD**.
- Registro: REG-FUNC. Identificado por 01. É ítem de grupo pois é composto por vários subgrupos.
- DADOS PESSOAIS: também é ítem de grupo pois é composto por vários subgrupos.
- ENDEREÇO: também é ítem de grupo pois é composto por vários ítens elementares (RUA, NÚMERO, etc.).
- Repare que à medida que os dados se subdividem, seus números de nível devem aumentar. É de costume utilizarmos somente os números ímpares (os pares também podem ser utilizados).
- Os números de nível utilizados na descrição dos registros estão compreendidos no intervalo de 01 a 49.

Ainda sobre o exemplo acima, os ítens DIA, MES e ANO aparecem duas vezes, em DATA-NASCIMENTO e DATA-ADMISSÃO. Para diferenciá-los devemos utilizar um *qualificador* (**IN** ou **OF** indistintamente). Assim, para referenciar ao dia de nascimento do funcionário usaríamos DIA OF DATA-NASCIMENTO e para referenciar ao dia de admissão do funcionário na empresa usaríamos DIA OF DATA-ADMISSÃO, possibilitando diferenciá-los.

Além dos números de nível para registros existem ainda três níveis especiais utilizados para dados que não pertencem à estrutura de registro. São eles:

- 66 : para grupos ou ítem elementares descritos pela cláusula **RENAME.** Não será discutido aqui.

- 77 : designa ítens indivisíveis da **WORKING-STORAGE SECTION.**

- 88 : especificam condições associadas a valores particulares.

*Exemplo:*

```
03  COR-DE-CABELO PIC 9.
    88 CASTANHA  VALUE  IS 1.
    88 NEGRA     VALUE  IS 2.
    88 LOURA     VALUE  IS 3.
    88 RUIVA     VALUE  IS 4.
```

### 5.1.2) Cláusula PICTURE

É através dessa cláusula que se descreve um campo atribuindo-lhe tipo, tamanho e informações especiais de edição.

Pode aparecer abreviada apenas por **PIC.**

- Tipos de dados:

  - Alfabético: identificado pelo caracter A (PIC A).

  - Numérico: identificado pelo dígito 9 (PIC 9). Tamanho máximo: 18 caracteres.

  - Alfanumérico: identificado pelo caracter X (PIC X).

O tamanho de um campo é identificado de duas formas possíveis. A primeira delas seria repetir o caracter que identifica o tipo do campo tantas vezes quantos forem os caracteres constituintes do campo. Este tipo de descrição não pode exceder a 3Ø caracteres.

*Exemplo:* campo CEP - numérico de 5 posições.
representação – CEP PIC 99999.

OBS - 99999 NÃO É O VALOR DO CEP!

A segunda forma seria identificar o tipo de campo através do caracter correspondente e ao seu lado escrevermos o número de posições que o campo ocupa entre parênteses.

*Exemplo:* campo CEP – numérico de 5 posições.
representação – CEP PIC 9 ( 5 )
tipo tamanho

*Outros exemplos:*

| CAMPO | TIPO | TAM. | REPRESENTAÇÕES |
|---|---|---|---|
| Placa-de-carro | Alfanumérico | Ø7 | PIC XXXXXXX ou PIC X(07) |
| Idade | Numérico | 03 | PIC 999 ou PIC 9(03) |
| Num-Telefone | Numérico | Ø7 | PIC 9999999 ou PIC 9(07) |
| Código-Tamanho | Alfabético | 01 | PIC A ou PIC A(01) |

- outros símbolos:

  - V: usado na indicação de um ponto decimal. Somente aparece em campos numéricos (claro!). Não é contado no tamanho do ítem.

    *Exemplo:* SALÁRIO PIC 999999V99
    ou
    PIC 9 (6) V99
    ou
    PIC 9 (6) V9(2)

  - S: indica que o ítem (numérico) possui um sinal (negativo ou positivo). Se omitirmos o S, o ítem é considerado SEMPRE positivo. Não é levado em conta no tamanho do ítem.

    *Exemplo:* DEFICIT PIC S9(5)V9(2)

  - caracteres de edição: para formatos especiais em relatórios impressos.

    - Ø: inserção de zeros.
    - $: inserção de $ (ou sinal corrente) à esquerda do valor.
    - B: inserção de brancos.

    etc.

### 5.1.3) A entrada FD

A entrada **FD** serve para descrevermos as características de todos os arquivos mencionados em cláusula **SELECT**. Para cada arquivo a ser descrito faz-se necessário o uso de uma **FD**. Portanto, a **FILE SECTION** terá tantas **FD's** quantos forem os arquivos a serem definidos.

A descrição do arquivo inicia-se com **FD** e também termina com um ponto (.) colocado logo após a última cláusula descritiva.

FORMATO GERAL:

| A | B |
|---|---|
| FD | *nome-do-arquivo*<br>LABEL RECORD . . .<br>VALUE OF FILE-ID . . .<br>BLOCK CONTAINS . . .<br>RECORD CONTAINS . . .<br>DATA RECORD IS . . . . |

↑ última cláusula descritiva

↑ ponto

- **LABEL RECORD:** esta cláusula deve estar sempre presente. Ela especifica se o arquivo deve ou não possuir registros especiais de identificação no seu início e no seu final, chamados *etiquetas* (vide Cap. 1).

FORMATO: **LABEL RECORD IS** [ **OMMITED** | **STANDARD** ]

Para arquivos de saída associados à impressora usa-se

**LABEL RECORD IS OMMITED**

que não atribui etiquetas ao arquivo.

Para arquivos associados a discos magnéticos usa-se

**LABEL RECORD IS STANDARD**

que atribui etiquetas ao arquivo.

- **VALUE OF FILE-ID:** usada apenas para arquivos em disco magnético. Especifica qual o nome que o arquivo terá no diretório do disco (nome-externo). Ela existe porque é possível associarmos a um arquivo um outro nome, um nome interno, que só serve para nos referirmos a um arquivo durante o programa e não precisa ser necessariamente o nome contido no diretório.

FORMATO: **VALUE OF FILE-ID-IS** *"d: nome-externo. extensão"*

onde:

d: unidade de disco onde encontra-se o arquivo.

nome-externo: é o nome do arquivo que consta no diretório.

extensão: é a extensão do nome. Em geral caracteriza o tipo de arquivo.

O nome externo deve possuir no máximo 8 caracteres e a extensão no máximo 3 caracteres.

- **BLOCK CONTAINS:** especifica para fins de leitura ou gravação, o tamanho de um *registro-físico* contido no arquivo.

FORMATO: **BLOCK CONTAINS** *número-inteiro* [**RECORDS** / **CHARACTERS**]

Ou seja, podemos especificar o tamanho de um *registro-físico* em termos do número de *registros-lógicos* ou em termos do número de caracteres.

Quando utilizamos a opção RECORDS, o *número-inteiro* que aparece nesta cláusula é chamado *fator de bloco*. Esta cláusula não é obrigatória.

*Exemplo:* 1) **BLOCK CONTAINS 10 RECORDS**

Especifica o fator de bloco. Se soubermos o número de caracteres do registro lógico é possível determinarmos o número de caracteres do registro físico.

| TAMANHO DO REG. LÓGICO x FATOR DE BLOCO = TAMANHO DO REG. FÍSICO |
|---|

2) **BLOCK CONTAINS 200 CHARACTERS**

Especifica o tamanho de um registro físico em número de caracteres.

- **RECORD CONTAINS:** especifica o tamanho do registro lógico (em número de caracteres).

FORMATO: **RECORD CONTAINS** número-inteiro1 **TO** número-inteiro2 **CHARACTERS.**

Esta cláusula não é obrigatória, já que o número de caracteres do registro pode ser calculado somando-se o número de caracteres de cada um de seus campos constituintes. No entanto, ela facilita a documentação do programa.

*Exemplo:* 1) **RECORD CONTAINS 80 CHARACTERS**

Neste caso os registros possuem tamanho fixo (80 caracteres).

2) **RECORD CONTAINS 100 TO 120 CHARACTERS**

Este arquivo hipotético contém registros de tamanho variável, sendo o menor de 100 posições e o maior de 120 posições.

- **DATA RECORD:** atribui um nome ao registro de um arquivo.

FORMATO: **DATA** [**RECORD IS** *nome-registro* / **RECORDS ARE** *nome-registro 1, nome-registro 2, . . .*]

Se utilizarmos mais de um nome significa que o arquivo possui mais de um tipo de registro e portanto devemos descrever cada um deles.

*Exemplo:*

```
A  | B
FD | arquivo 1
   |   .
   |   .
   |   .
   |   DATA RECORDS ARE nome-registro 1
   |   nome-registro2.
01 |   nome-registro 1.
   |         .
   |         .
   |         .
01 |   nome-registro 2.
   |
   |
```

*Exemplo:* descrever os seguintes arquivos:

1) Cadastro de alunos.
   - Gravado em disco.
   - Nome interno: CAD-ALU.
   - Nome externo: ALUNOS. TXT (TXT = arquivo de dados).
   - Fator de bloco = 10.
   - Nome do registro: REG-ALU.
   - Campos:

| CAMPO | TIPO | TAMANHO |
|---|---|---|
| Nome | Alfanumérico | 40 |
| Endereço | Alfanumérico | 40 |
| Curso | Alfabético | 05 |
| Sala | Numérico | 02 |

2) Relatório de alunos reprovados.
   - Saída em impressora de 80 colunas.
   - Nome interno: ALUN-REP.
   - Nome do registro: REG-REP.

Micro News

# Folha de Codificação Cobol

PAGINA 01 / 01

| | | | |
|---|---|---|---|
| AUTOR: EDUARDO | N.o DE CARTÕES: — | N.o DO PROTOCOLO | Autor: EDUARDO |
| SISTEMA / CURSO: EXEMPLO / MICRO NEWS | N.o DE FOLHAS: 01 | | Identificação 73 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

| Número de Sequencia (1 2 3 4 5 6) | CONT (7) | A (8 9 10 11) | B (12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71) |
|---|---|---|---|
| 010 | | | |
| 020 | | | |
| 030 | | DATA | DIVISION. |
| 040 | | FILE | SECTION. |
| 050 | | FD | CAD-ALU |
| 060 | | | LABEL RECORD IS STANDARD |
| 070 | | | VALUE OF FILE-ID IS "ALUNOS.TXT" |
| 080 | | | BLOCK CONTAINS 10 RECORDS |
| 090 | | | RECORD CONTAINS 87 CHARACTERS |
| 100 | | | DATA RECORD IS REG-ALUN. |
| 110 | | 01 | REG-ALU. |
| 120 | | | 03 NOME PIC X(40). |
| 130 | | | 03 ENDERECO PIC X(40). |
| 140 | | | 03 CURSO PIC A(05). |
| 150 | | | 03 SALA PIC 9(02). |
| 160 | | FD | ALUN-REP |
| 170 | | | LABEL RECORD IS OMITTED |
| 180 | | | DATA RECORD IS REG-REP. |
| 190 | | 01 | REG-REP. |
| 200 | | | 03 REG-REP PIC X(80). |

## 5.2) WORKING-STORAGE SECTION

Utilizada para descrever campos auxiliares de trabalho associados ao nível 77 e também para a definição de linhas de relatório impresso.

### 5.2.1) Cláusula FILLER

Representa uma área a qual não será atribuida um nome. Representa também a ocorrência de *brancos* ou *zeros* conforme seja alfanumérica ou numérica.

*Exemplo:*

```
03 FILLER  PIC X (10) .        10 brancos

03 FILLER  PIC 9 (05) .        5 zeros
```

Alguns compiladores permitem o uso da opção SIZE (tamanho).

*Exemplo:* 03 FILLER SIZE IS 5. 5 brancos.

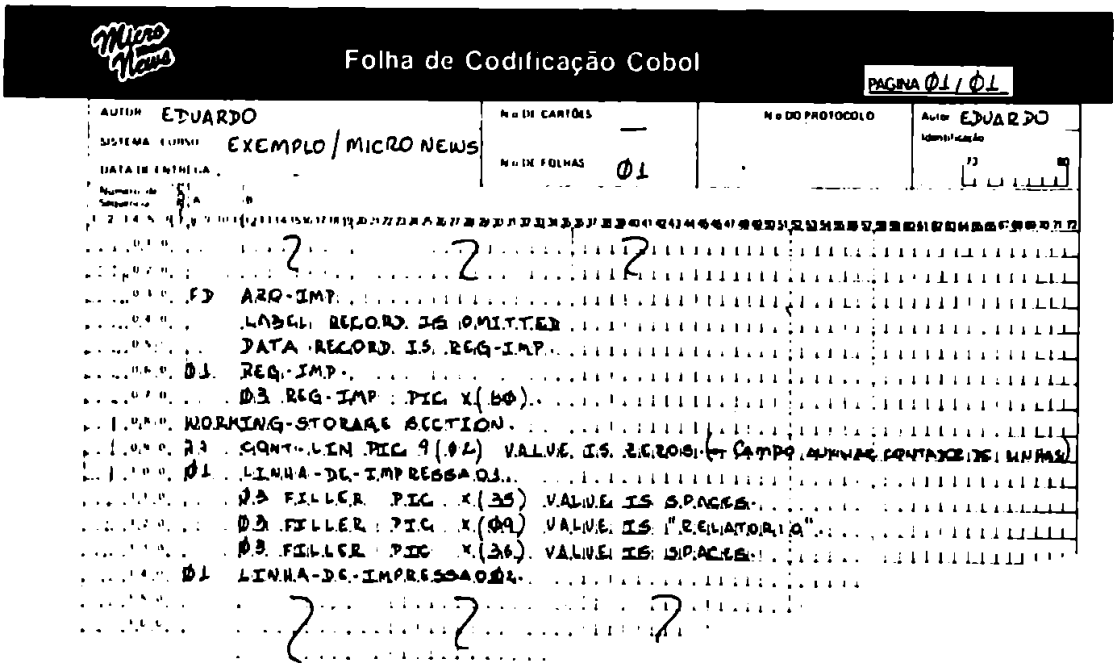
### 5.2.2) Cláusula VALUE

Atribui um valor (valor inicial) a um campo ou uma constante.

*Exemplo:*

```
03 FILLER PIC 9(5) VALUE IS "04070.

03 FILLER PIC X(40) VALUE IS SPACES.
```




No exemplo acima vimos a construção de uma linha de relatório impresso. Essa linha é uma área auxiliar do registro de impressão REG-IMP e é acessada pelo comando **WRITE FROM** na **PROCEDURE DIVISION.**

```
WRITE REG-IMP FROM LINHA-DE-IMPRESSÃO1.
```

Espaço reservado para anotações de aula.

Espaço reservado para anotações de aula.

# CAPÍTULO 6

# PROCEDURE DIVISION

— COMANDO DISPLAY
— COMANDO ACCEPT
— COMANDO MOVE
— COMANDO GO TO
— COMANDO IF
— COMANDO STOP
— DEFINIÇÃO DE TELAS
— TRATAMENTO DE RELATÓRIOS
— ARQUIVOS SEQUENCIAIS
— ARQUIVOS INDEXADOS
— TRATAMENTO DE RELATÓRIOS A NÍVEL DE PROGRAMA
— CLASSIFICAÇÃO DE ARQUIVOS
— COMANDOS ARITMÉTICOS
— TABELAS
— COMANDO PERFORM

## 6) PROCEDURE DIVISION

É a quarta divisão do COBOL.

Contém os comandos executáveis do programa. Possui apenas seções definidas pelo próprio programador. Seus comandos são agrupados em parágrafos.

### 6.1) Comando DISPLAY

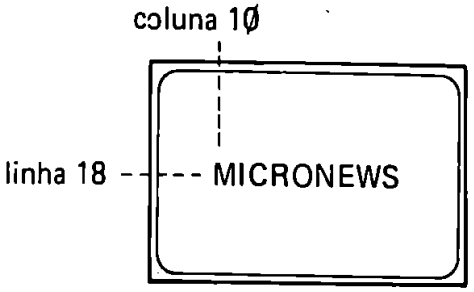
Usado para saída de informações (Lembram-se do PRINT e do LPRINT?).

FORMATO:
- **DISPLAY (L, ♭C)** *mensagem ou campo.*
- **DISPLAY** *mensagem ou campo* **UPON** *mnemônico.*

No primeiro formato (correspondente à saída em vídeo):

- L: linha de vídeo onde deverá ser colocada a mensagem (1 a 24).
- C: coluna onde deverá começar a mensagem ( 1 a 8Ø).

*Exemplo:* DISPLAY (12,♭1Ø) "MICRONEWS".




No segundo formato, o uso da opção UPON significa que a saída de dados será via impressora. O *mnemônico* deverá estar definido no **SPECIAL NAME PRINTER IS** *mnemônico.*

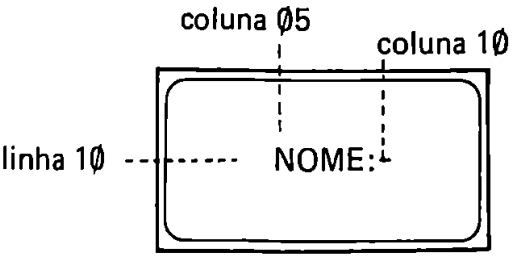
OBS.: Não se esqueçam que os **SPECIAL-NAMES** são definidos na **CONFIGURATION SECTION.**

### 6.2) Comando ACCEPT

Permite a entrada de dados via teclado.

FORMATO: **ACCEPT (L, ♭C)** *nome-do-campo.*

*Exemplo:* **DISPLAY (1Ø♭Ø5)** "NOME:".
**ACCEPT (1Ø♭1Ø)** NOME.




OBS.: Nos dois comandos anteriores, o uso da vírgula entre linha e coluna é facultativo, no entanto o branco é obrigatório com ou sem vírgula.

### 6.3) Comando MOVE

Este comando permite a movimentação de valores de um campo para outro ou de

um determinado valor para um determinado campo.

FORMATO: **MOVE** *nome-de-campo1* **TO** *nome-de-campo2*.
**MOVE** *valor* **TO** *campo.*

O comando **MOVE** moverá o conteúdo do *campo1* para o *campo2* (sem que *campo1* altere seu valor) ou ainda moverá o *valor* para o campo determinado.

*Exemplo:* MOVE W77-DIA TO DIA-REL.
MOVE 88 TO AND REL.

6.4) Comando **GO TO**
Transfere a execução do programa de um parágrafo para outro.

FORMATO: **GO TO** *nome-do-parágrafo.*
**GO TO** *parag1, parag2, parag3* . . . **DEPENDING ON** *campo.*

No primeiro formato a execução é transferida para o parágrafo especificado.

*Exemplo:* GO TO INICIO

No segundo formato o fluxo é desviado para algum dos parágrafos especificados dependendo do valor do campo.

*Exemplo:* GO TO PAR-A, PAR-B, PAR-C DEPENDING ON CONTADOR.

| Conteúdo do campo CONTADOR | Desvio para |
|---|---|
| 1 | PAR-A |
| 2 | PAR-B |
| 3 | PAR-C |
| qualquer outro valor | comando seguinte |

6.5) Comando **IF**
Uso semelhante ao já conhecido do BASIC. Estabelece condições associadas a comandos.

FORMATO: **IF** *condição comando1* **ELSE** *comando2.*

- condição verdadeira — execução do comando1 e em seguida próximo comando após o **IF.**

- condição falsa — se a cláusula **ELSE** for explicitada, execução do comando2. Se a cláusula **ELSE** não for explicitada, execução do comando seguinte ao **IF.**

*Exemplo:* IF CAMPO1 = 5Ø MOVE Ø TO CAMPO1
ELSE COMPUTE CAMPO1 = CAMPO1 + 1.
↖
fim do parágrafo

Neste comando também pode ser usada a opção **NEXT SENTENCE** que passa o controle do processamento para o comando que vem a seguir ao **IF.**
Existem vários tipos de testes que o comando **IF** pode fazer. Alguns deles:

- teste de nome de condição.

Usado para verificar se o valor de um campo é ou não igual aos valores associados ao *nome-de-condição.*

*Exemplo:*
```
03 ESTADO-CIVIL PIC 9.
   88 SOLTEIRO VALUE IS 1.
   88 CASADO VALUE IS 2.
   88 VIUVO VALUE IS 3.
   88 DIVORCIADO VALUE IS 4.
   88 TICO-TICO-NO-FUBA VALUE IS 5.
   88 ERRO VALUE IS 6 THRU 9.
                      ↑
                     até

IF SOLTEIRO GO TO PAR-1.
IF CASADO   GO TO PAR-2.
```

- Teste de sinal.

Usado para verificar se o valor de um campo é positivo, negativo ou nulo.

IF *campo* **IS (NOT)** — [ **POSITIVE** / **NEGATIVE** / **ZERO** ]

*Exemplo:* IF DEFICIT IS NOT NEGATIVE RES = "LUCRO".

- Teste de relação.

Estabelece uma comparação entre dois valores utilizando um *operador relacional.* Os operadores relacionais encontram-se no quadro abaixo.

| OPERADOR | SIGNIFICADO |
|---|---|
| EQUAL TO OU = | Igual |
| LESS THAN OU < | Menor |
| GREATHER THAN OU > | Maior |
| NOT EQUAL TO OU NOT = | Diferente |
| NOT LESS THAN OU NOT < | Maior ou Igual |
| NOT GREATHER + THAN OU NOT > | Menor ou Igual |

OBS.: Não são permitidos os símbolos $\geqslant$ e $\leqslant$ para maior ou igual e menor ou igual respectivamente.

*Exemplo:*
```
IF VALOR IS GREATHER THAN 50 COMPUTE VALOR = 0
                             ELSE
                             COMPUTE VALOR = VALOR + 1
```

### 6.6) Comando STOP

Suspende ou finaliza a execução de um programa.

FORMATO: **STOP** [ **RUN** / *literal* ]

No primeiro formato, temos o encerramento do programa.

No segundo formato a *literal* é exibida em vídeo e a execução está suspensa até a intervenção do operador via teclado.

### 6.7) Definição de telas

As telas a serem utilizadas durante a execução do programa são definidas na **SCREEN SECTION ( DATA DIVISION )**.

Este ítem foi colocado neste capítulo para que pudéssemos exemplicar o uso de telas na **PROCEDURE DIVISION.**

A **SCREEN SECTION** é a quarta seção da **DATA DIVISION** e é utilizada na definição de formatos de vídeo. É composta por entradas de descrições de vídeo no mesmo formato da **FILE SECTION** e da **WORKING-STORAGE SECTION.**

As descrições podem ser agrupadas através da atribuição de números aos níveis apropriados.

— O *número de nível* deve ser um inteiro de 1 a 49.

— O *nome do vídeo* deve concordar com as regras de formatação de nomes.

#### 6.7.1) Funções executáveis da SCREEN SECTION

- **BLANK SCREEN:** apaga o vídeo todo e situa o cursor em (Ø1 Ø1).

- **LINE e COLUMN:** indicam uma posição no vídeo. Se não forem especificadas a posição atual não será alterada. Se **COLUMN** é especificada sem **LINE**, a linha não é ajustada. Se **LINE** é especificada sem **COLUMN** assume-se a coluna 1.

- **VALUE IS:** especifica a cadeia de caracteres a ser emitida. Esta cadeia deve estar delimitada por aspas.

- **PICTURE (PIC):** formato através do qual os dados serão exibidos.

- **TO:** é opcional. Durante um **ACCEPT** o sistema movimenta o dado a qualquer campo.

- **FROM:** também é opcional. Utilizado em **DISPLAY** quando há a movimentação de qualquer campo ou literal para um ítem temporário. O conteúdo do ítem temporário é exibido em vídeo.

- **BLANK WHEN ZERO:** faz com que o conteúdo de um campo de valor zero seja exibido como espaço em branco.

*Exemplo:*

## Folha de Codificação Cobol

PAGINA 01 / 02

AUTOR: Eduardo / Toninho
SISTEMA / CURSO: Exemplo / Micro News
DATA DE ENTREGA: ___ / ___ / ___
N.o DE CARTÕES: —
N.o DE FOLHAS: 02
N.o DO PROTOCOLO
Autor: Eduardo/Toninho
Identificação 73 80

| Número de Sequência | A | B |
|---|---|---|
| 010 | | IDENTIFICATION DIVISION. |
| 020 | | |
| 030 | | |
| 040 | FD | DEMO |
| 050 | 01 | REG-DEM. |
| 060 | | 02 COD-DEM. |
| 070 | | 05 NUME-DEMO PIC 9(05). |
| 080 | | 02 NOME-DEMO PIC X(30) |
| 090 | | 02 ENDE-DEMO PIC X(40) |
| 100 | | |
| 110 | WORKING-STORAGE SECTION. | |
| 120 | 77 | W77-FSDEMO PIC X(02) VALUE SPACES. |
| 130 | | |
| 140 | | |
| 150 | SCREEN SECTION. | |
| 160 | 01 | TELA-DEMO. |
| 170 | | 02 BLANK SCREEN. |
| 180 | | 02 LINE 01 COLUMN 01 VALUE IS |
| 190 | | "TELA MODELO - VERSAO 1.00". |
| 200 | | 02 LINE 03 COLUMN 01 VALUE IS |

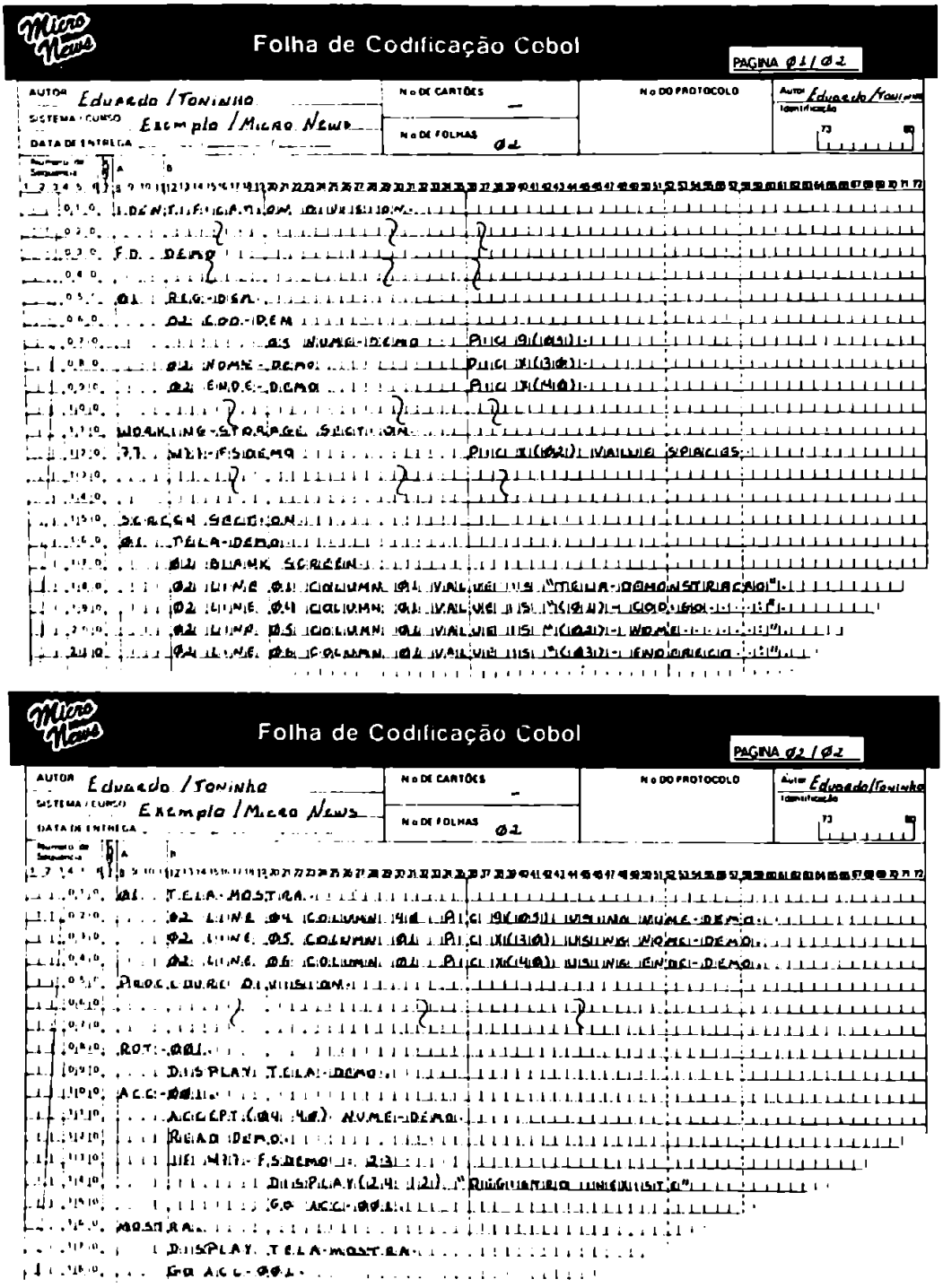
## Folha de Codificação Cobol

PAGINA 02 / 02

AUTOR: Eduardo / Toninho
SISTEMA / CURSO: Exemplo / Micro News
DATA DE ENTREGA: ___ / ___ / ___
N.o DE CARTÕES: —
N.o DE FOLHAS: 02
N.o DO PROTOCOLO
Autor: Eduardo/Toninho
Identificação 73 80

| Número de Sequência | A | B |
|---|---|---|
| 010 | | "DEMONSTRACAO DE TELA DE SISTEMA". |
| 020 | | 02 LINE 04 COLUMN 01 VALUE IS |
| 030 | | "(01)- CODIGO.....:". |
| 040 | | 02 LINE 05 COLUMN 01 VALUE IS |
| 050 | | "(02)- NOME.....:". |
| 060 | | 02 LINE 06 COLUMN 01 VALUE IS |
| 070 | | "(03)- ENDERECO.....:". |
| 080 | | |
| 090 | | |
| 100 | PROCEDURE DIVISION. | |
| 110 | INICIO. | |
| 120 | | |
| 130 | | |
| 140 | ROT-001 | |
| 150 | | DISPLAY TELA-DEMO. |
| 160 | ACC-001. | |
| 170 | | ACCEPT (04 40) NUME-DEMO. |
| 180 | | ACCEPT (05 40) NOME-DEMO. |
| 190 | | ACCEPT (06 40) ENDE-DEMO. |
| 200 | GRAVA. | |

No exemplo acima, vimos como usar uma formatação de tela para entrada de dados. Abaixo veremos uma tela para mostrar dados de um arquivo através de seus campos.




## 6.8) Tratamento de relatórios

Um relatório é basicamente formado por três comandos:

1 - **MOVE.**

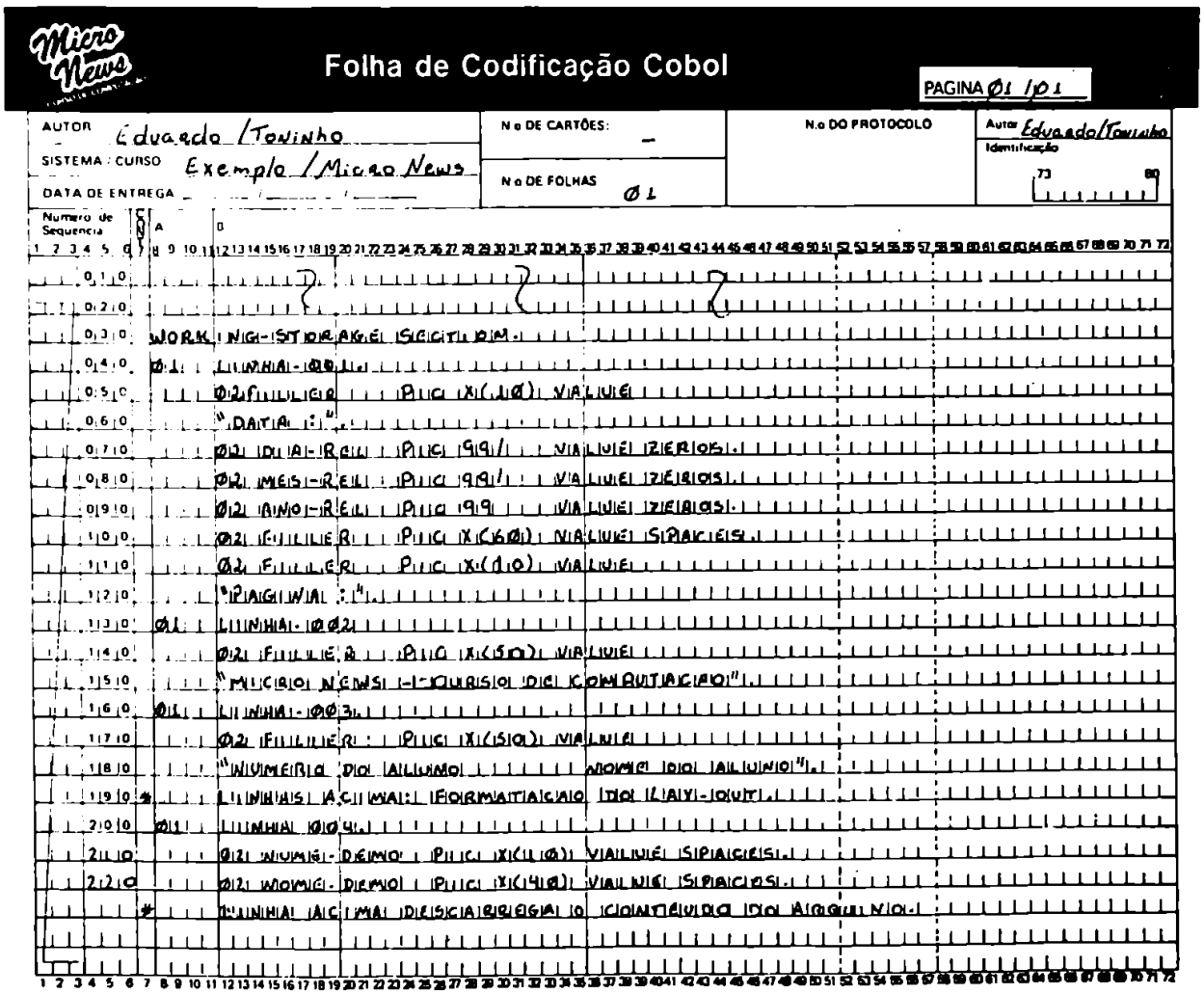
2 - **WRITE.**

3 - **FILLER.**

Para um relatório no formato abaixo:

| Data: 99/99/99 | MICRO NEWS - Curso de Computação | Página: 999 |
|---|---|---|
| Número do aluno | Nome do aluno | |
| X-.....X | X.....X | |

As linhas de impressão seriam descritas na **WORKING-STORAGE SECTION** da seguinte maneira.




Mais a frente veremos como utilizar esse exemplo do comando **FILLER** na parte de arquivos indexados.

## 6.9) Arquivos sequenciais

### 6.9.1) Comando OPEN

Utilizado na abertura de um arquivo. Dependendo do tipo de operação a que o arquivo será submetido, a comando **OPEN** é utilizado com alguma das opções abaixo:

```
          ┌ INPUT  . . . . . leitura
          │ OUTPUT . . . . . gravação
OPEN ─────┤ I-O    . . . . . leitura e gravação
          └ EXTEND . . . . . inclusão de registros
```

```
                ┌ INPUT  ┐
                │ OUTPUT │
FORMATO: OPEN ──┤ I-O    ├── nome-do-arquivo.
                └ EXTEND ┘
```

OBS.: Só podem estar abertos simultaneamente durante a execução do programa um máximo de 14 arquivos.

### 6.9.2) Comando CLOSE

Fecha um arquivo.

FORMATO: **CLOSE** *nome-do-arquivo.*

### 6.9.3) Comando READ (leitura de um arquivo sequencial)

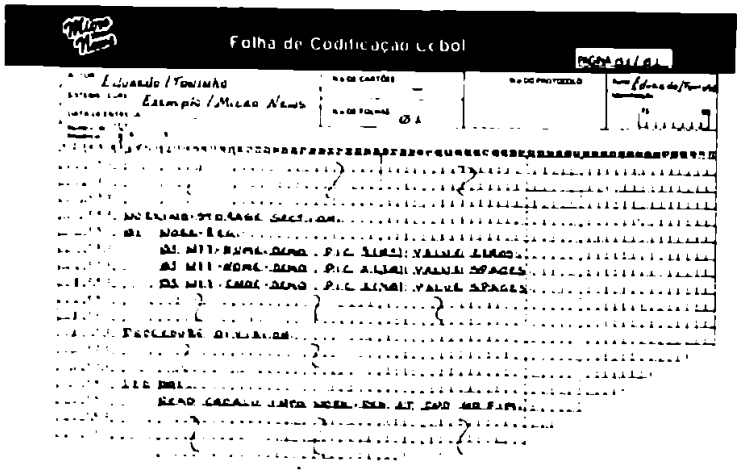
Serve para ler registros que estão gravados em um arquivo.

FORMATO: **READ** *nome-do-arquivo* **INTO** *área* **AT END** *comando imperativo*

Uma característica importante dos arquivos com organização sequencial é que para lermos um determinado registro é necessário que se tenha lido todos os antecessores.

- Cláusula **INTO:** permite que o conteúdo de um determinado registro seja movimentado automaticamente para uma área especificada.

*Exemplo:*



O conteúdo do registro será movimentado para a área auxiliar WORK-REG que se encontra na **WORKING-STORAGE SECTION.**

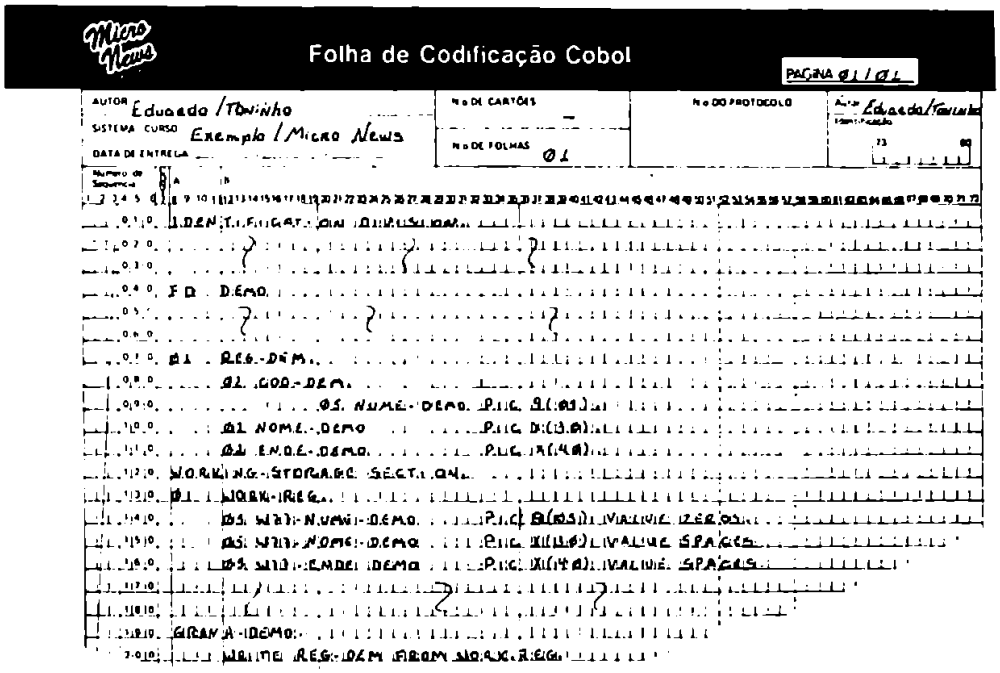
Após a leitura do último registro será executado o comando que estiver em frente à cláusula **AT END.**

### 6.9.4) Comando WRITE (gravação em arquivo sequencial)

Serve para "gravar" registros em arquivos de saída.

FORMATO: **WRITE** *nome-do-registro.*
**WRITE** *nome-do-registro* **FROM** *área.*

- Cláusula **FROM:** permite que se grave um conteúdo qualquer que se encontra armazenado em uma determinada área sem que seja necessário a movimentação campo a campo. Move-se o registro inteiro para a área desejada como no exemplo abaixo:




### 6.9.5) Comando REWRITE (atualização de arquivo)

Utilizado na atualização de registros. Só pode ser utilizado em arquivos abertos como **OPEN I-O** já que a atualização envolve o processo de leitura e gravação.

FORMATO: **REWRITE** *nome-do-registro.*
**REWRITE** *nome-do-registro* **FROM** *área.*

### 6.10) Arquivos indexados

Os arquivos indexados têm como característica principal permitir o acesso a registros aleatóriamente, ou seja, não é preciso ler os antecessores, indo-se direto ao registro desejado.

Da mesma forma não é necessário preocupar-se com a ordem em que os registros estão gravados pois eles estarão sempre armazenados em ordem segundo algum de seus campos constituintes.

Para tanto, os arquivos indexados mantém um diretório particular de índices (chaves). São esses índices que permitem que os *ponteiros* acessem diretamente os registros.

*Ponteiro*, em outras palavras, vem a ser um campo cujo conteúdo é o endereço de onde se encontra localizada determinada informação dentro do arquivo.

Como já dissemos, os arquivos (sequenciais e indexados) são gerados ordenadamente de acordo com algum campo que compõe seus registros. No caso dos arquivos indexados estes campos são denominados *chaves* e é através dessas *chaves* que podemos ler, gravar, alterar ou deletar (apagar) registros diretamente.

Essas *chaves* além de constarem dos *registros de dados*, compõe juntamente com os *ponteiros* o *controle dos índices*. Esse controle não deve preocupar-nos pois quem se encarrega de manipulá-los é o próprio *sistema operacional*, sem que seja necessária qualquer intervenção de nossa parte.

### 6.10.1) Implicações na cláusula SELECT

Os arquivos indexados exigem a inclusão de algumas opções no assinalamento (SELECT) do arquivo.

```
SELECT arquivo  ASSIGN TO DISK
                ORGANIZATION IS INDEXED
                ACCESS MODE IS  SEQUENCIAL
                                DYNAMIC
                                RANDOM
                RECORD KEY IS nome-de-campo
                FILE STATUS IS nome-de-campo.
```

- **ORGANIZATION IS INDEXED**

Para que o programa consiga identificar que o arquivo a ser manipulado é indexado é necessário declarar esta opção na SELECT.

- **ACCESS MODE IS . . .**

**. . . SEQUENCIAL**

Apesar de um arquivo ter sido gerado (organizado) de maneira indexada é possível acessá-lo sequencialmente. Este método de acesso é escolhido geralmente para ler ou gravar um arquivo inteiro, ou ainda, a partir de um determinado ponto desse arquivo.

**. . . DYNAMIC**

Este modo de acesso é escolhido para ler ou gravar registros tanto sequencialmente como randomicamente.

**. . . RANDOM**

Este modo de acesso é escolhido para ler ou gravar registros aleatóriamente, ou seja, em pontos distintos do arquivo, de maneira direta.

- **RECORD KEY IS** *nome-de-campo*

Nesta cláusula deve ser declarada a *chave* do arquivo (nome-de-campo) obrigatóriamente definida no LAY-OUT do registro na FD.

OBS.: O tamanho máximo da chave é de 6Ø bytes.

- **FILE STATUS IS** *nome-de-campo*

Na operação de arquivos indexados é obrigatório toda vez que se faz qualquer tipo de manipulação de dados, testar se a operação foi bem sucedida ou não. Este teste será feito através do *campo* que colocarmos nesta cláusula. Este campo deverá, ainda, ser definido na **WORKING-STORAGE SECTION** como campo alfanumérico de duas posições **(PIC X ( Ø2).**

Assim sendo, toda vez que se fizer algum tipo de manipulação de dados, o sistema devolverá um código ao campo **FILE STATUS.** De posse desse valor, o programa toma as medidas necessárias ao auxílio do processamento.

A seguir encontra-se um quadro com os possíveis conteúdos do **FILE STATUS.** É importante notar que pelo fato do **FILE STATUS** possuir duas posições alfanuméricas, o quadro apresentará o diagnóstico do caracter da esquerda (vertical), que corresponde ao tipo de problema ocorrido ou se a operação foi bem sucedida, enquanto o caracter da direita (horizontal) apresenta o motivo do problema.

| CARACTER ESQUERDO | CARACTER DIREITO | | | | |
|---|---|---|---|---|---|
| | SEM (0) PROBLEMAS | ERRO DE (1) ESTRUTURA | CHAVE (2) DUPLICADA | NENHUM RE-GIST.ENC.(3) | ESPAÇO DE(4) DISCO ESG. |
| COMPLETADO COM SUCESSO (0) | X | | | | |
| AT END (1) | X | | | | |
| INVALID KEY (2) | | X | X | .X | X |
| ERRO PERMANENTE (3) | X | | | | X |
| CASOS ESPECIAIS (9) | | X | | | |

### 6.10.2) Abertura de arquivos indexados

Com excessão da opção EXTEND, a abertura de arquivos indexados obedece às mesmas regras de abertura já vistas no comando **OPEN.**

A cláusula **EXTEND** não se aplica a arquivos indexados.

*Exemplo:*

# Folha de Codificação Cobol

PAGINA 01 / 01

AUTOR: Eduardo / Toninho

SISTEMA / CURSO: Exemplo / Micro News

DATA DE ENTREGA: ___ / ___ / ___

Nº DE CARTÕES: —

Nº DE FOLHAS: 01

Nº DO PROTOCOLO:

Autor: Eduardo/Toninho

Identificação: 73 — 80

| Número de Sequência | CONT | A | B |
|---|---|---|---|
| 010 | | | |
| 020 | | | |
| 030 | | | |
| 040 | | PROCEDURE DIVISION. | |
| 050 | | INICIO. | |
| 060 | | | OPEN I-O DEMO. |
| 070 | | | IF W77-FSDEMO = 00 NEXT SENTENCE |
| 080 | | | ELSE |
| 090 | | | IF W77-FSDEMO = 30 OPEN OUTPUT DEMO |
| 100 | | | CLOSE DEMO GO ABRIR |
| 110 | | | ELSE |
| 120 | | | DISPLAY(L C) "ERRO DE ABERTURA" |
| 130 | | | CLOSE DEMO |
| 140 | | | STOP RUN. |
| 150 | | ROT-001 | |
| 160 | | | |
| 170 | | | |
| 180 | | | |
| 190 | | | |
| 200 | | | |

No exemplo acima mostramos como deverá ser feita a rotina de abertura de um arquivo indexado.

O modo de abertura **I-O ( INPUT-OUTPUT )** indica ao COBOL que o mesmo executará rotinas de leitura e gravação sem que se feche o arquivo para cada operação distinta.

### 6.1Ø.3) Posicionamento de arquivos indexados

Posicionar um arquivo significa colocá-lo em ordem de acordo com suas necessidades.

*Exemplo:*

Em um determinado arquivo encontram-se mais ou menos 1ØØØ registros referentes a alunos. Vamos supor que só nos interessa um relatório a partir do aluno de código 7ØØ. Se a organização do arquivo não for indexada, a listagem começaria no aluno de código 1, já com organização indexada, a listagem poderia começar de onde quisermos.

FORMATO: **MOVE LOW-VALUES TO** *nome-de-campo*
**MOVE** { **ZERDS** / **SPACES** } **TO** *nome-de-campo*
**START** *nome-do-arquivo* **KEY IS** { **GREATHER** / **NOT LESS** / **EQUAL** } *nome-de-dado*

Em outras palavras, posicionamos esse arquivo em um determinado ponto para que a leitura seja realizada. Esse posicionamento só pode ser utilizado em arquivos de acesso sequencial ou dinâmico cobertos com **OPEN I-O** ou **OPEN INPUT**. O *nome-de-campo* deve ser a chave de acesso do arquivo e o valor a ser checado no arquivo deve ser fornecido em comando imediatamente anterior ao **START.**

### 6.1Ø.4) Leitura de arquivos indexados

- **ACCESS MODE IS SEQUENTIAL**

FORMATO: **READ** *nome-do-arquivo* **INTO** *nome-de-dado*

A leitura para o acesso sequencial de organização indexada funciona de maneira igual à concebida até o momento.

A cláusula **INTO** é opcional.

O final do arquivo deverá ser detectado através do código 1Ø do **FILE STATUS.**

Pode-se utilizar o comando **START**, quando então, o arquivo será lido a partir da chave posicionada.

- **ACCESS MODE IS RANDOM**

FORMATO: **READ** *nome-de-arquivo* **INTO** *nome-de-dado*

A leitura para acesso randômico é semelhante aos métodos vistos anteriormente, podendo-se de acordo com a conveniência utilizar a cláusula **INTO.**

Como o acesso é aleatório, é possível alcançar pontos distintos do arquivo de maneira direta, sendo que antes de efetuar-se o comando **READ** é necessário mover-se o valor referente à chave do registro requerido.

- **ACCESS MODE IS DYNAMIC**

FORMATO: **READ** *nome-do-arquivo* **NEXT RECORD** *nome-de-dado*

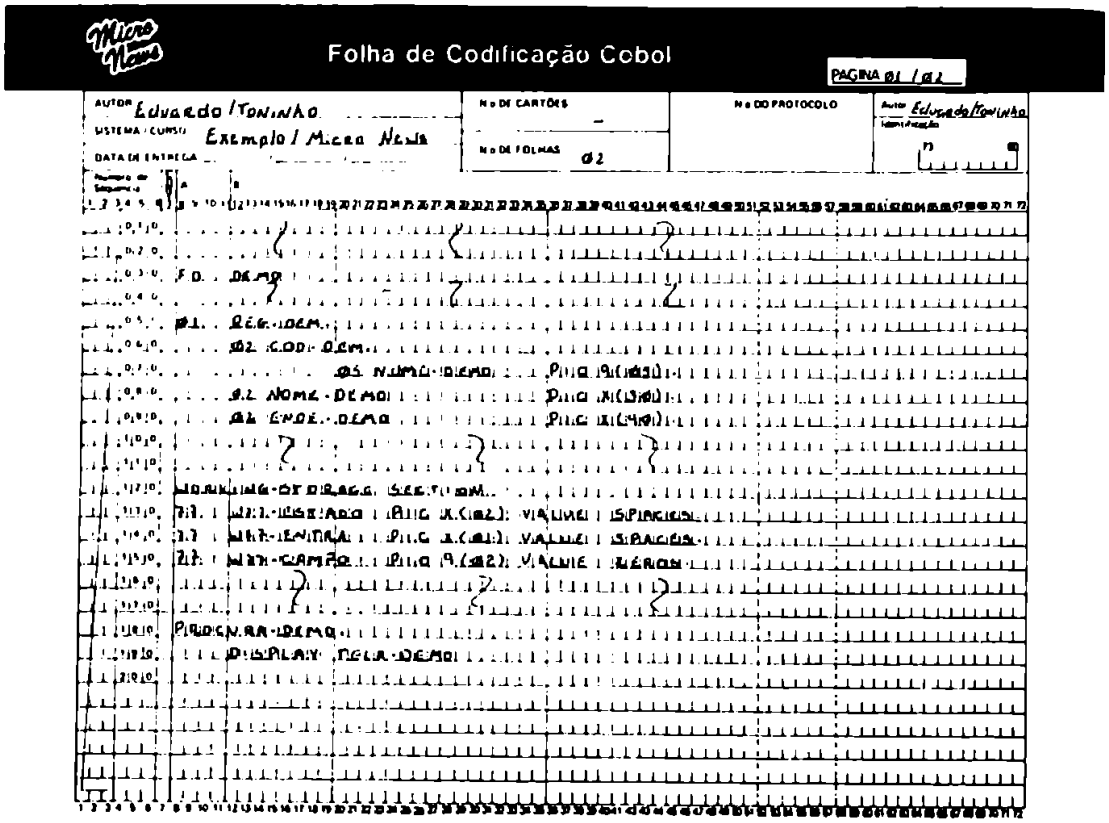
Permite que se leia um arquivo de maneira sequencial ou randômica durante a mesma execução do programa.

Em caso de leitura sequencial, deve-se definir através do comando **START** o ponteiro do qual deseja-se ler. O final do arquivo será detectado por **FILE STATUS** igual a 10 (código de fim de arquivo).

Para a leitura randômica, deve-se proceder de maneira análoga ao acesso randômico.

### 6.10.5) Gravação em arquivo indexado

Para podermos gravar em um arquivo de organização indexada devemos em primeiro lugar pesquisar se a chave já existe. Caso não exista, podemos utilizar o seguinte procedimento:




No exemplo acima procuramos a chave por **INVALID KEY** que nos traz um retorno automático dizendo se a chave existe ou não.

Não podemos trabalhar com chaves duplicadas, pois seria equivalente a procurar determinado assunto em um livro com dois números iguais de páginas.

Note ainda que verificamos a cada gravação se o controle de nosso processamento é igual a 00 ( **FILE STATUS** ) que indica sucesso na operação.

Micro News
CURSO DE COMPUTAÇÃO

# Folha de Codificação Cobol

PAGINA 02 / 02

AUTOR: Eduardo / Toninho
SISTEMA / CURSO: Exemplo / Micro News
DATA DE ENTREGA: ___ / ___ / ___

N.o DE CARTÕES: —
N.o DE FOLHAS: 02

N.o DO PROTOCOLO

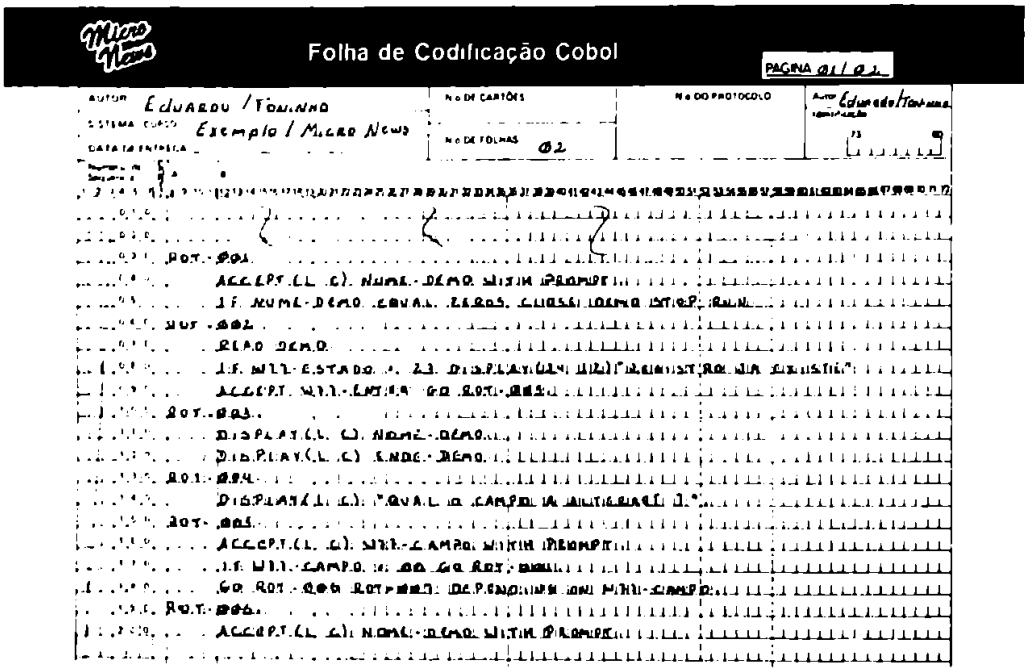
Autor: Eduardo/Toninho
Identificação: 73 80

| Número de Sequencia (1–6) | CONT (7) | A (8–11) | B (12–35) | (36–72) |
|---|---|---|---|---|
| 010 | | ROT-001. | | |
| 020 | | | ACCEPT(L C) NUME-DEMO WITH PROMPT. | |
| 030 | * | | PROMPT PROMOVE A EXIBICAO DE UMA ESTEIRA DE PONTOS. | |
| 040 | | | IF NUME-DEMO EQUAL ZEROS | |
| 050 | | | CLOSE DEMO | |
| 060 | | | STOP RUN. | |
| 070 | | ROT-002. | | |
| 080 | | | READ DEMO INVALID KEY GO ROT-003. | |
| 090 | | | DISPLAY(24 12) "REGISTRO JA EXISTE" | |
| 100 | | | ACCEPT W77-ENTRA GO ROT-004. | |
| 110 | | ROT-003. | | |
| 120 | | | ACCEPT(L C) NOME-DEMO WITH PROMPT. | |
| 130 | | ROT-004. | | |
| 140 | | | ACCEPT(L C) ENDE-DEMO WITH PROMPT. | |
| 150 | | GRAV | A-DEMO. | |
| 160 | | | WRITE REG-DEM. | |
| 170 | | | IF W77-ESTADO = 00 NEXT SENTENCE | |
| 180 | | | ELSE | |
| 190 | | | DISPLAY(24 12) "ERRO DE GRAVACAO" | |
| 200 | | | W77-ESTADO | |
| 210 | | | CLOSE DEMO | |
| 220 | | | STOP RUN. | |

| FILE STATUS IMPORTANTES ( GUARDE-OS BEM) ! | |
|---|---|
| 10 | Fim de Arquivo |
| 22 | Chave duplicada (registro já existe) |
| 23 | Chave não existe |
| 24 | Disco cheio (área onde estão sendo gravados os dados não comporta mais informações |
| 30 | Arquivo inexistente (a área onde deveriam estar os dados não consta) |
| 91 | Erro de estrutura (arquivo danificado ou aberto) |

### 6.10.6) Atualização de arquivos indexados

No exemplo abaixo mostraremos como regravar em um arquivo indexado.




Foi utilizado em linhas gerais o mesmo procedimento de inclusão de registros com duas pequenas alterações:

1 - Ao invés de testarmos por **INVALID KEY**, testamos por **FILE STATUS** a fim de verificar se existe a chave procurada para alteração dos dados. É importante salientar que não podemos alterar a chave de acesso à informação. Tome por exemplo um livro. Para localizarmos um determinado assunto vamos direto ao índice e verificamos em que página se encontra.

2 - Ao invés de usarmos **WRITE** usamos **REWRITE** pois o registro já foi gravado uma vez. Como queremos somente mudar a área de dados, simplesmente redigitamos em cima do próprio campo.

# Folha de Codificação Cobol

PAGINA Ø2 / Ø2

| | | | |
|---|---|---|---|
| AUTOR: Eduardo / Toninho | N.o DE CARTÕES: – | N.o DO PROTOCOLO | Autor: Eduardo/Toninho |
| SISTEMA / CURSO: Exemplo / Micro News | N.o DE FOLHAS: Ø2 | | Identificação 73 — 80 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

```
Número de   CONT  A    B
Sequência
 010              ROT-007.
 020                   ACCEPT (L C) END-DEMO WITH PROMPT.
 030              REGRAVA-DEMO.
 040                   REWRITE REG-DEMO.
 050                   IF W33-ESTADO = 00 NEXT SENTENCE
 060                   ELSE
 070                   DISPLAY (24 12) "ERRO DE REGRAVACAO"
 080                   W33-ESTADO
 090                   CLOSE DEMO
 100                   STOP RUN.
 110
 120
 130
 140
 150
 160
 170
 180
 190
 200
```

OBS.: A cláusula **DEPENDING ON** foi vista anteriormente. Esperamos que seu uso não cause nenhum tipo de dificuldade.

### 6.10.7) Exclusão de registros de arquivos indexados

A organização indexada permite eliminar registros do arquivo através do comando **DELETE.**

FORMATO: **DELETE** *nome-do-arquivo*

Em caso de acesso sequencial o último comando relacionado ao arquivo deve ser um **READ** bem sucedido, portanto o último registro será o registro a ser deletado, não havendo necessidade de se testar **FILE STATUS = 23.**

Para o acesso randômico não é necessária a leitura do registro a ser deletado. Caso não exista o registro correspondente ao conteúdo da chave, o programa sairá por **FILE STATUS** igual a 23.

*Exemplo:*

1 - Acesso Randômico:

```
MOVE W77-CHAVE TO NUME-DEM.
DELETE DEMO.
IF W77-ESTADO = ØØ NEXT SENTENCE
ELSE
DISPLAY (L C) "ERRO DE DELEÇÃO"
W77-ESTADO CLOSE DEMO
STOP RUN.
```

2 - Acesso Indexado:

```
DELETE DEMO.
IF W77-ESTADO = ØØ NEXT SENTENCE
ELSE
DISPLAY (L C) "ERRO DE DELEÇÃO"
W77-ESTADO CLOSE DEMO
STOP RUN.
```

### 6.11) Tratamento de relatórios a nível de programa

Basicamente adapta-se a **SELECT** conforme mostrado abaixo.

Micro News

## Folha de Codificação Cobol

PAGINA 01 / 04

| AUTOR Eduardo Toninho | Nº DE CARTÕES: — | Nº DO PROTOCOLO | Autor: Eduardo Toninho Identificação |
|---|---|---|---|
| SISTEMA / CURSO Exemplo / Micro News | Nº DE FOLHAS: 04 | | |
| DATA DE ENTREGA ___ / ___ / ___ | | | |

Número de Sequência | A | B

```
000100
000200
000300
000400 SELECT CADASTRO ASSIGN TO DISK
000500         ORGANIZATION INDEXED
000600         ACCESS MODE DYNAMIC
000700         RECORD KEY COD-ARQ
000800         FILE STATUS W-ERRO.
000900 SELECT RELATORIO ASSIGN TO PRINTER.
001000
001100
001200 FD  CADASTRO
001300 LABEL RECORD STANDARD
001400 VALUE OF FILE-ID "CADASTRO.DAT".
001500 01 REG-ARQ.
001600    02 COD-ARQ      PIC 9(06).
001700    02 NOME-ARQ     PIC X(30).
001800    02 NUMERO-ARQ   PIC 9(06).
001900    02 NOME-ARQ     PIC X(30).
002000
002100 FD RELATORIO
002200 LABEL RECORD OMITTED.
002300 01 REGISTRO PIC X(80).
002400 WORKING-STORAGE SECTION.
002500 77 W-ERRO PIC X(02) VALUE "00".
002600 77 W-CONT PIC 9(04) VALUE ZEROS.
002700 77 W-OPCAO PIC X(01) VALUE SPACES.
002800 77 W-ACT PIC 9(02) VALUE ZEROS.
002900 77 MENS PIC X(50) VALUE SPACES.
003000 77 LIMPA PIC X(50) VALUE SPACES.
003100 01 CAB0.
003200    02 FILLER PIC X(20) VALUE SPACES.
003300    02 FILLER PIC X(40) VALUE
003400    "RELATORIO DE CLIENTES".
003500 01 CAB1.
003600    02 FILLER PIC X(06) VALUE "CODIGO".
003700    02 FILLER PIC X(02) VALUE SPACES.
003800    02 FILLER PIC X(30) VALUE "NOME".
003900    02 FILLER PIC X(02) VALUE SPACES.
004000    02 FILLER PIC X(06) VALUE "NUMERO".
004100    02 FILLER PIC X(02) VALUE SPACES.
004200    02 FILLER PIC X(30) VALUE "NOME".
004300 01 DET.
004400    02 DET-CODIGO PIC 9(06) VALUE ZEROS.
004500    02 FILLER PIC X(02) VALUE SPACES.
004600    02 DET-NOME PIC X(30) VALUE SPACES.
004700    02 FILLER PIC X(02) VALUE SPACES.
004800    02 DET-NUMERO PIC 9(06) VALUE ZEROS.
004900    02 FILLER PIC X(02) VALUE SPACES.
005000    02 DET-NOME PIC X(30) VALUE SPACES.
005100
```

Micro News

## Folha de Codificação Cobol

PAGINA 02 / 04

| AUTOR: Eduardo Toninho | Nº DE CARTÕES: — | Nº DO PROTOCOLO | Autor: Eduardo Toninho Identificação |
|---|---|---|---|
| SISTEMA / CURSO: Exemplo / Micro News | Nº DE FOLHAS: 04 | | |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

Número de Sequência | A | B

```
000100 FD  RELATORIO
000200 LABEL RECORD OMITTED
000300 01 REGISTRO-IMP PIC X(80).
000400
000500
000600 PROCEDURE DIVISION.
000700 INICIO.
000800 OPEN I-O CADASTRO
000900 IF W-ERRO NOT = "00"
001000    IF W-ERRO = "30"
001100       OPEN OUTPUT CADASTRO
001200       CLOSE CADASTRO
001300       MOVE "ARQUIVO CADASTRO SENDO CRIADO" TO MENS
001400       PERFORM ROT-MENS THRU ROT-MENS-FIM
001500       GO TO INICIO
001600    ELSE
001700       MOVE "ERRO NA ABERTURA DO ARQUIVO CADASTRO" TO MENS
001800       PERFORM ROT-MENS THRU ROT-MENS-FIM
001900       GO TO ROT-FIM.
002000 OPEN OUTPUT RELATORIO
002100 MOVE ZEROS TO COD-ARQ.
002200 START CADASTRO KEY NOT LESS COD-ARQ INVALID KEY
002300       MOVE "ARQUIVO CADASTRO VAZIO" TO MENS
002400       PERFORM ROT-MENS THRU ROT-MENS-FIM
002500       GO TO ROT-FIM.
002600 WRITE REGISTRO-IMP FROM CAB0.
002700 WRITE REGISTRO-IMP FROM CAB1.
002800 LER-CADASTRO.
002900 READ CADASTRO NEXT AT END GO TO ROT-FIM.
003000 IF W-ERRO NOT = "00"
003100    MOVE "ERRO NA LEITURA DO ARQUIVO CADASTRO" TO MENS
003200    PERFORM ROT-MENS THRU ROT-MENS-FIM
003300    GO TO ROT-FIM.
003400 MOVE COD-ARQ TO DET-CODIGO.
003500 MOVE NOME-ARQ TO DET-NOME.
003600 MOVE NUMERO-ARQ TO DET-NUMERO.
003700 MOVE NOME-ARQ TO DET-NOME.
003800 WRITE REGISTRO-IMP FROM DET.
003900 ADD 1 TO W-CONT.
004000 IF W-CONT < 20 GO TO LER-CADASTRO.
004100 MOVE ZEROS TO W-CONT.
004200 MOVE "PRESSIONE ENTER PARA CONTINUAR" TO MENS.
004300 PERFORM ROT-MENS THRU ROT-MENS-FIM.
004400 GO TO LER-CADASTRO.
004500 ROT-FIM.
004600 CLOSE CADASTRO RELATORIO.
004700 STOP RUN.
004800 ROT-MENS.
004900 MOVE ZEROS TO W-CONT.
005000 DISPLAY (23, 12) MENS.
005100 ROT-MENS-FIM.
005200 EXIT.
```

Micro News

# Folha de Codificação Cobol

PAGINA 03 / 04

AUTOR: Eduardo / Toninho

SISTEMA / CURSO: Exemplo / Micro News

DATA DE ENTREGA: ____ / ____ / ____

N.o DE CARTÕES: —

N.o DE FOLHAS: 04

N.o DO PROTOCOLO

Autor: Eduardo/Toninho

Identificação 73 80

| Número de Sequencia (1 2 3 4 5 6) | CONT (7) | A (8 9 10 11) | B (12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72) |
|---|---|---|---|
| 010 | | 01 | LINHA-002- |
| 020 | | | 02 FILLER PIC X(50) VALUE |
| 030 | | | "MICRO NEWS - CURSO DE COMPUTACAO". |
| 040 | | 01 | LINHA-003- |
| 050 | | | 02 FILLER PIC X(50) VALUE |
| 060 | | | "NUMERO DO ALUNO NOME DO ALUNO". |
| 070 | | 01 | LINHA-004- |
| 080 | | | 02 NUME-ALU PIC X(10) VALUE SPACES. |
| 090 | | | 02 NOME-ALU PIC X(40) VALUE SPACES. |
| 100 | | SCREEN SECTION. | |
| 110 | | 01 | TELA-MOSTRA. |
| 120 | | 0 | 02 BLANK SCREEN. |
| 130 | | | 02 LINE 01 COLUMN 01 VALUE |
| 140 | | | "MICRO NEWS - CURSO DE COMPUTACAO". |
| 150 | | | 02 LINE 02 COLUMN 01 VALUE |
| 160 | | | "EMISSAO DE RELATORIO DE CHAMADA". |
| 170 | | | 02 LINE 10 COLUMN 10 |
| 180 | | | "CONFIRMA IMPRESSAO...[S/N]". |
| 190 | | | |
| 200 | | | |

# Folha de Codificação Cobol

PAGINA Ø4 / Ø4

AUTOR: Eduardo / Toninho | N.o DE CARTÕES: — | N.o DO PROTOCOLO | Autor: Eduardo/Toninho

SISTEMA / CURSO: Exemplo / Micro News | N.o DE FOLHAS: Ø4 | Identificação 73 – 80

DATA DE ENTREGA: ____ / ____ / ____

```
Número de
Sequencia   A    B
010
020
030        MOSTRA-TELA.
040            DISPLAY TELA-MOSTRA.
050            ACCEPT (1Ø 4Ø) W33-CONFIRMA WITH PROMPT.
060            IF W33-CONFIRMA = "S" OR EQUAL "s" GO ROT-ØØ1.
070            IF W33-CONFIRMA = "N" OR EQUAL "n" GO ROT-FIM.
080        LER-ØØ1.
090            READ CADALU NEXT RECORD AT END GO ROT-ØØ4.
100            MOVE NUME-ALU TO NUME-RE.
110            MOVE NOME-ALU TO NOME-RE.
120        IMPRIME.
130            ADD 1 TO W33-PAG.
140            MOVE W33-PAG TO PAG-RE.
150            WRITE LINHA FROM LINHA-ØØ1.
160            WRITE LINHA FROM LINHA-ØØ2.
170            WRITE LINHA FROM LINHA-ØØ3.
180
190
200
```

No exemplo acima vimos como podemos relacionar os comandos **WRITE, FILLER** e **MOVE**, sendo que o comando WRITE, já utilizado na gravação de informações, foi utilizado aqui para impressão na forma de relatório.

### 6.12) Classificação de arquivos

Antes de fornecermos a receita do **SORT**, devemos citar o que é um **SORT. SORT** é uma classificação de um arquivo.

Pode ser uma classificação interna (intrínseca) ou externa.

A classificação interna é a que utiliza o **SORT** do COBOL e a externa necessita de um arquivo auxiliar ao programa COBOL para que nele sejam feitas as operações (independente do tamanho da memória) necessárias à conclusão do trabalho.

Esse arquivo auxiliar é o arquivo **SORT**. Para ele devem ser definidas quase todas as cláusulas e comandos de manuseio de um arquivo comum. Sua filosofia como arquivo, porém, é um pouco diferente da usual. Esse arquivo tem sua área automaticamente definida quando ele for requisitado pelo programa e tem a mesma desprezada pelo sistema automaticamente ao término da operação.

Portanto, um arquivo de classificação não poderá ser usado a não ser pelo programa que o classificou.

- Diferenças na codificação e estrutura.

Para um arquivo de SORT define-se:

| A | B |
|---|---|
| | **SELECT** *nome-do-arquivo-de-SORT* ASSIGN TO DISK |
| | SORT STATUS IS *campo-numérico.* |
| | " |
| SD | *nome-do-arquivo-de-SORT* |
| | " |
| Ø1 | *nome-do-registro.* |
| | Ø3 . . . |

O campo numérico deverá ser definido na **WORKING-STORAGE SECTION** com duas posições ( **PIC 9(2)** ).

Note que para um arquivo SORT a definição começa com SD e não FD. O arquivo SORT deverá ser previamente descrito na **SELECT.**

Dado que o programa tem um SORT interno que irá classificar determinados registros de um arquivo, durante seu processamento pode surgir a necessidade de se fazer algum procedimento específico antes ou mesmo após a classificação. Por isso, o formato do SORT para a **PROCEDURE DIVISION** torna-se:

**SORT** *nome-do-arquivo* **ON** [**ASCENDING** / **DESCENDING**] **KEY** *chave-de-classificação*

**INPUT PROCEDURE** *nome-do-parágrafo1* **(SECTION)**

**OUTPUT PROCEDURE** *nome-do-parágrafo2* **(SECTION)**

ou

**USING** *nome-de-arquivo1*
**GIVING** *nome-de-arquivo2*

- *nome-de-arquivo:* é o nome do arquivo auxiliar no qual será feita a classificação.

- **ASCENDING / DESCENDING:** especifica-se a classificação da chave do arquivo: crescente ou decrescente.

- *chave-de-classificação:* especifica qual o campo ou quais os campos do registro de SORT serão chaves pelas quais será classificado o arquivo. A classificação será feita nos campos na ordem em que eles forem declarados.

*Exemplo:* SORT ARQ-SORT ASCENDING KEY NUME-SRT NOME-SRT

O programa irá classificar o ARQ-SORT de forma crescente em relação às chaves especificadas (NUME-SRT e NOME-SRT). Se tivéssemos definido no registro de arquivo os dois campos acima como ítens elementares como por exemplo CHAVE-DEM como ítem de grupo, teríamos o seguinte comando:

SORT ARQ-SORT ASCENDING KEY CHAVE-DEM

A partir desse formato teríamos duas opções que de certa forma poderiam ser usadas ao mesmo tempo.

- Opção 1:

**INPUT PROCEDURE** *nome-do-parágrafo1* **(SECTION)**
**OUTPUT PROCEDURE** *nome-do-parágrafo2* **(SECTION)**

Esta opção indica que todos os procedimentos a serem tomados antes da classificação do arquivo devem estar localizados em *nome-do-parágrafo1*, que obrigatoriamente deverá ter uma **SECTION** da **PROCEDURE DIVISION**. Indica também que após a classificação do arquivo os procedimentos necessários devem estar especificados em *nome-do-parágrafo2*, que deverá também ter uma SECTION contendo os comandos requisitados.

O COBOL ao encontrar esse tipo de comando agirá da seguinte maneira: ao deparar-se com o comando **SORT**, ele irá executar a **INPUT PROCEDURE** e todos os comandos que fazem parte da mesma. Após o término da execução irá devolver a **OUTPUT PROCEDURE**, que por sua vez fará o mesmo. Logo após a execução das duas SECTION's o controle será devolvido ao programa.

- Opção 2:

**USING** *nome-de-arquivo1*
**GIVING** *nome-de-arquivo2*

Esta opção fornece ao COBOL todos os dados necessários para a classificação, pois ao declarar-se **USING**, define-se qual o arquivo a ser classificado. Esse arquivo será automaticamente aberto, lido e gravado no arquivo auxiliar para classificação e logo após fechado.

Ao definir-se **GIVING** define-se qual o arquivo que receberá o conteúdo classificado durante a execução do SORT. Para tanto ele deverá abrir o arquivo saída, lê-lo e posteriormente gravá-lo no arquivo de SORT.

Deve-se alertar contudo, que as operações com **OPEN** ou **CLOSE** no caso dos dois arquivos são desnecessárias pois isso é feito automaticamente.

Para gravar o arquivo SORT, usamos o comando **RELEASE**. O mesmo será gravado na forma sequencial.

FORMATO: **RELEASE** *nome-do-registro-de-SORT* **FROM** *área-auxiliar.*

A opção **FROM** é utilizada caso estejamos usando uma determinada área de trabalho da **WORKING-STORAGE SECTION** para guardar dados a serem gravados, ou mesmo para o caso de entrada do arquivo a ser classificado externamente como o arquivo SORT.

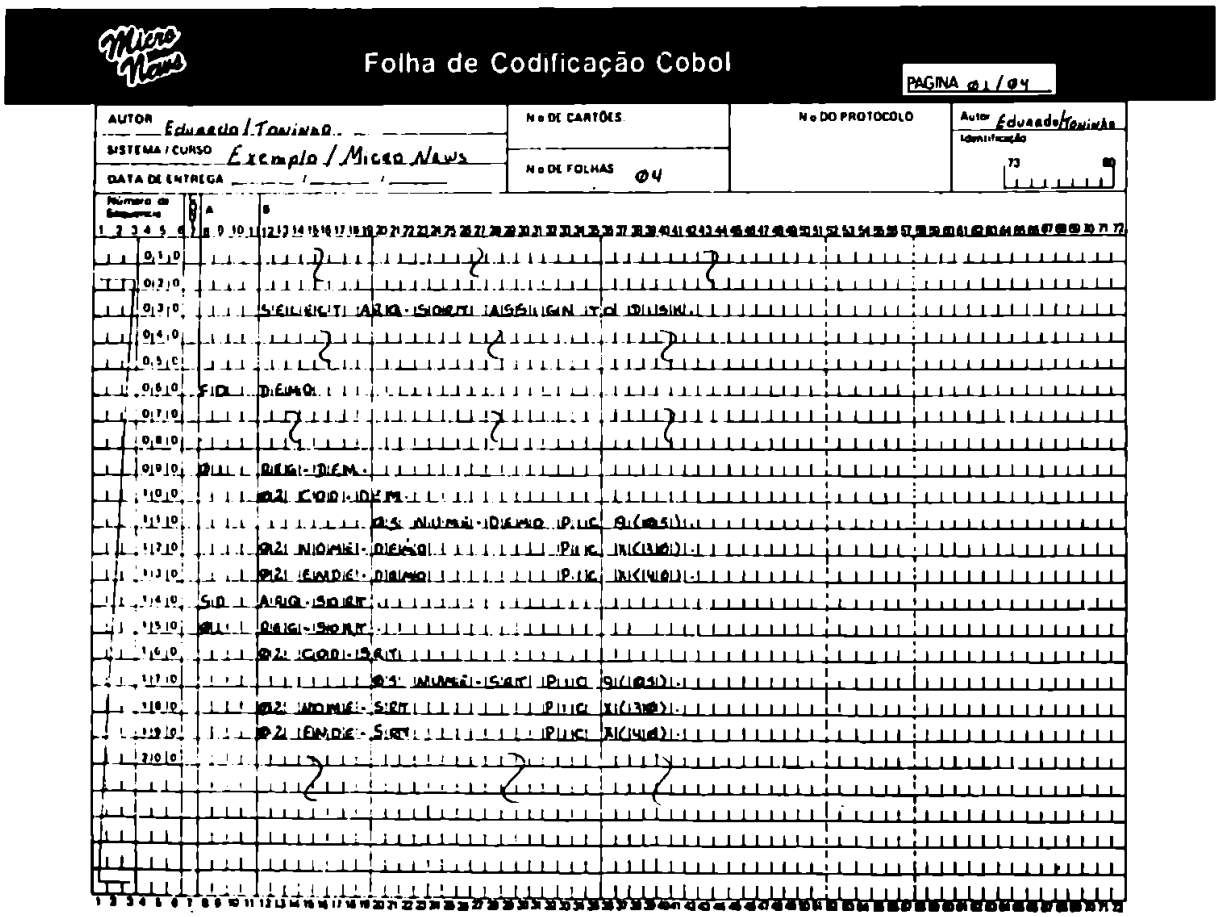
Para ler-se um arquivo SORT usamos o comando **RETURN**. Ele engloba todas as características de um comando **READ** em um arquivo sequencial.

FORMATO: **RETURN** *nome-do-arquivo* **INTO** *área-auxiliar* **AT END** *comando*

A opção é usada analogamente à opção **FROM**. Utilizamos a área auxiliar da **WORKING-STORAGE SECTION** ou até mesmo a área de outro arquivo.

A opção **AT END** é uma codificação de final de leitura de um arquivo. A partir do momento que o final venha a ser detectado, ele sairá após o primeiro ponto (.) decontrado.

*Exemplo:*



# Folha de Codificação Cobol

PAGINA 02 / 04

AUTOR: Eduardo / Toninho

SISTEMA / CURSO: Exemplo / Micro News

DATA DE ENTREGA: ____ / ____ / ____

N.o DE CARTÕES:

N.o DE FOLHAS: 04

N.o DO PROTOCOLO

Autor: Eduardo/Toninho

Identificação 73 80

| Número de Sequencia | CONT | A | B |
|---|---|---|---|
| 1 2 3 4 5 6 | 7 | 8 9 10 11 | 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 |
| 010 | | WORK | ING-STORAGE SECTION. |
| 020 | | 77 | W77-FIM PIC X(02) VALUE SPACES. |
| 030 | | SCREEN | SECTION. |
| 040 | | 01 | TELA-DEMO. |
| 050 | | 02 | BLANK SCREEN. |
| 060 | | 02 | LINE 01 COLUMN 01 VALUE " |
| 070 | | " | TELA-DEMONSTRACAO". |
| 080 | | 02 | LINE 04 COLUMN 01 VALUE " |
| 090 | | "(01) | CODIGO: ". |
| 100 | | 02 | LINE 05 COLUMN 01 VALUE " |
| 110 | | "(02) | NOME: ". |
| 120 | | 02 | LINE 06 COLUMN 01 VALUE " |
| 130 | | "(0131) | FINALIDADE: ". |
| 140 | | | |
| 150 | | | |
| 160 | | 01 | TELA-MOSTRA. |
| 170 | | 02 | LINE 09 COLUMN 40 PIC 9(05) USING NUME-DEMO. |
| 180 | | 02 | LINE 05 COLUMN 01 PIC X(30) USING NOME-DEMO. |
| 190 | | 02 | LINE 06 COLUMN 01 PIC X(40) USING COMP-DEMO. |
| 200 | | | |
| 210 | | | |
| 220 | | | |
| 230 | | | |
| 240 | | | |

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72

# Folha de Codificação Cobol

PAGINA 03/04

AUTOR Eduardo/Toninho
SISTEMA / CURSO Exemplo/Micro News
DATA DE ENTREGA ___/___/___
N.o DE CARTÕES:
N.o DE FOLHAS: 04
N.o DO PROTOCOLO
Autor: Eduardo/Toninho
Identificação

```
000100 PROCEDURE DIVISION.
000200 INICIO.
000300*
000400*
000500 ROT-000.
000600     DISPLAY TELA-DEMO.
000700 COMECA-SORT.
000800     SORT ARQSORT ON {ASCENDING} KEY COD-SRT
000900                     {DESCENDING}
001000     INPUT PROCEDURE ROT-000
001100     OUTPUT PROCEDURE ROT-002.
001200 SAIDA.
001300     CLOSE DEMO.
001400     STOP RUN.
001500 ROT-000 SECTION.
001600     READ DEMO AT END EXIT RECORD AT END GO FORA.
001700     RELEASE REG-SRT.
001800     GO LER-DEMO.
001900
002000
002100
002200
002300
002400
002500
002600
002700
002800
002900
003000
003100
003200
003300
003400
003500
003600
003700
003800
003900
004000
004100
004200
004300
004400
004500
004600
004700
004800
004900
005000
005100
005200
005300
005400
005500
005600
005700
005800
005900
006000
006100
006200
006300
006400
006500
006600
006700
006800
006900
007000
007100
007200
007300
007400
007500
007600
007700
007800
007900
008000
008100
008200
008300
008400
008500
008600
008700
008800
008900
009000
009100
009200
009300
009400
009500
009600
009700
009800
009900
010000
```

Micro News

# Folha de Codificação Cobol

PAGINA 04 / 04

AUTOR: Eduardo / Toninho

SISTEMA / CURSO: Exemplo / Micro News

DATA DE ENTREGA: ____ / ____ / ____

N.o DE CARTÕES:

N.o DE FOLHAS: 04

N.o DO PROTOCOLO

Autor: Eduardo Toninho

Identificação 73 80

| Número de Sequencia | CONT | A | B |
|---|---|---|---|
| 010 | | FORA. | |
| 020 | | EXIT. | |
| 030 | | ROT-002 SECTION. | |
| 040 | | LER-ARQ-SORT. | |
| 050 | | RETURN ARQ-SORT AT END GO FORA2. | |
| 060 | * | MOVIMENTACAO DOS CAMPOS PARA AREAS DE TRABALHO. | |
| 070 | | EXIT | |
| 080 | | | |
| 090 | | "2ª OPÇÃO" | |
| 100 | | | |
| 110 | | SORT ARQ-SORT ON | ASCENDING KEY COD-SRT |
| 120 | | | DESCENDING |
| 130 | | | USING ARQUIVO1 |
| 140 | | | GIVING ARQUIVO2. |
| 150 | | SAIDA. | |
| 160 | | STOP RUN. | |
| 170 | | | |
| 180 | | | |
| 190 | | | |
| 200 | | | |

### 6.13) Comandos aritméticos

O COBOL como uma linguagem puramente comercial possue também (como outras linguagens) comandos aritméticos. Neste trabalho, citaremos somente os mais utilizados.

- Operações

\+ ⟶ adição
\- ⟶ subtração
* ⟶ multiplicação
/ ⟶ divisão
** ⟶ exponenciação

OBS.: 1- Dois operadores aritméticos não podem aparecer juntos. Para evitar o encontro, fazemos uso dos parênteses.

2- O número de parênteses abertos " (" em uma expressão aritmética deve ser igual ao número de parênteses fechados")".

3- As operações aritméticas seguem a seguinte hierarquia:
   i) Operações contidas em parênteses (de dentro para fora).
   ii) Potenciação.
   iii) Multiplicação e divisão.
   iv) Adição e subtração.
   v) em caso de operações de mesma hierarquia é executada primeiramente a que estiver mais à esquerda.

- Comandos aritméticos (intrínsecos do COBOL)

  - **ADD:** soma um determinado valor em uma variável.

  *Exemplo:* ADD 2 TO W77-ACUMULA

  - **SUBTRACT:** subtrai de uma variável e pode ser utilizado juntamente com a opção GIVING, armazenando o valor em outra variável (o mesmo vale para o **ADD**).

  *Exemplo:* SUBTRACT 2 FROM W77-ACUMULA GIVING W77-TOTAL.

  - **COMPUTE:** usado na configuração de expressões matemáticas.

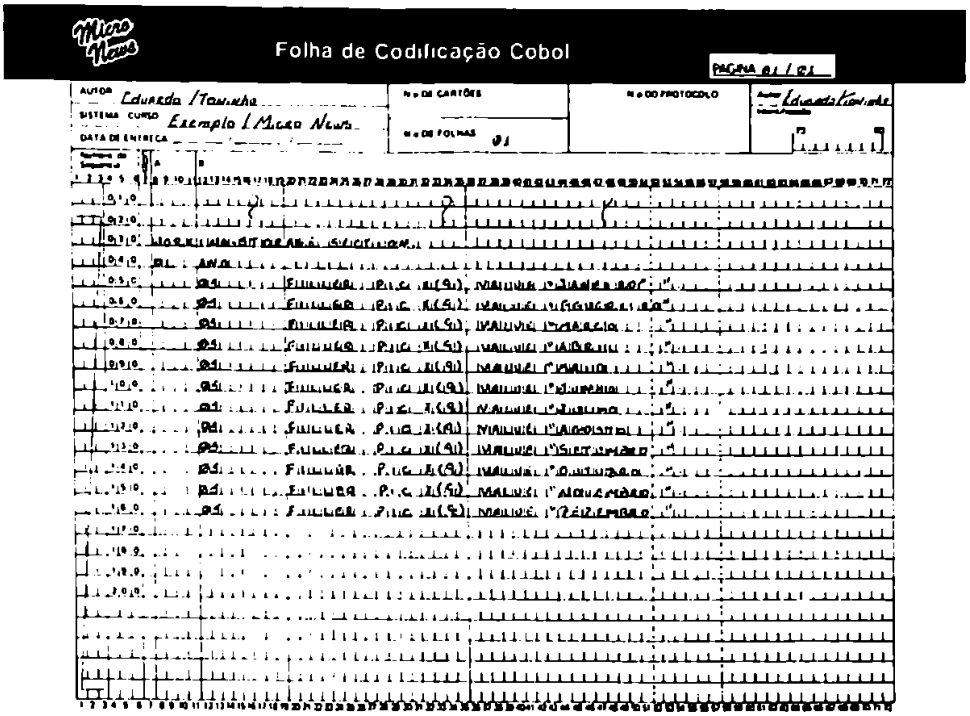
  *Exemplo:* COMPUTE W77-ACUMULA = W77-ACUMULA - W77-TOTAL.

Existem outros comandos aritméticos intrínsecos do COBOL tais como **MULTIPLY** e **DIVIDE.**

### 6.14) Tabelas

Para explicarmos a utilização de tabelas e seu manuseio, devemos antes dizer que uma tabela tem muito a ver com um registro, pois consiste em um nível maior e vários ítens elementares que se repetem algumas vezes.

Por exemplo, se tivéssemos um campo que tenha por valores os nomes de todos os meses do ano em seus subníveis, poderíamos definir:




Note que todos os ítens da tabela, são iguais (mesmo tamanho e mesmas características), portanto seria lógico tentarmos representar todos esses níveis dentro de um único campo.

### 6.14.1) Cláusula OCCURS

Ela nos permite dizer ao COBOL que um grupo (obrigatoriamente de nível diferente de Ø1) repete-se a partir dela tantas vezes quantas forem especificadas, mantendo as mesmas características mas com conteúdos (**VALUE**) distintos, ou mesmo não especificados.

FORMATO: *nível nome-do-campo* **OCCURS** *número de vezes* **TIMES**

ou **INDEXED BY** *índice*

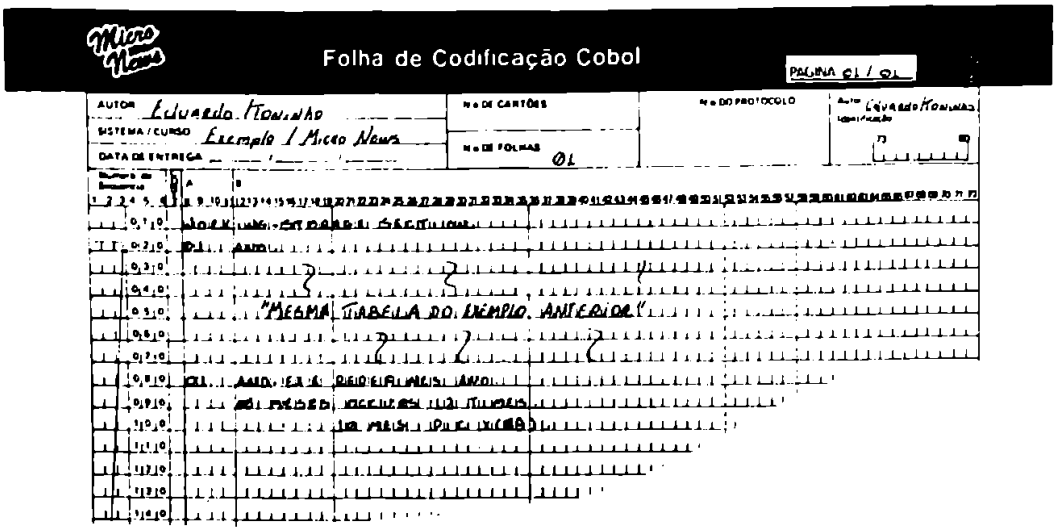
Aplicando essa cláusula teríamos:

```
Ø1 ANO.
   Ø5 MESES OCCURS 12 TIMES.
      1Ø MES PIC X (Ø9).
```

Note que foi dito ao COBOL apenas que o campo MESES tem 12 ocorrências e que possui 9 posições alfanuméricas. Nenhuma informação foi fornecida ao COBOL sobre o conteúdo dessa tabela. Para que as duas tabelas acima sejam iguais em termos de conteúdo de campo, temos duas opções:

1 - Movermos os valores para os campos durante a execução do programa na **PROCEDURE DIVISION.**

2 - Usarmos o conceito **REDEFINES** e acoplarmos os dois exemplos. Ou seja:




Dessa forma unimos o fato de termos um único campo com várias ocorrências e valores iniciais pré-determinados.

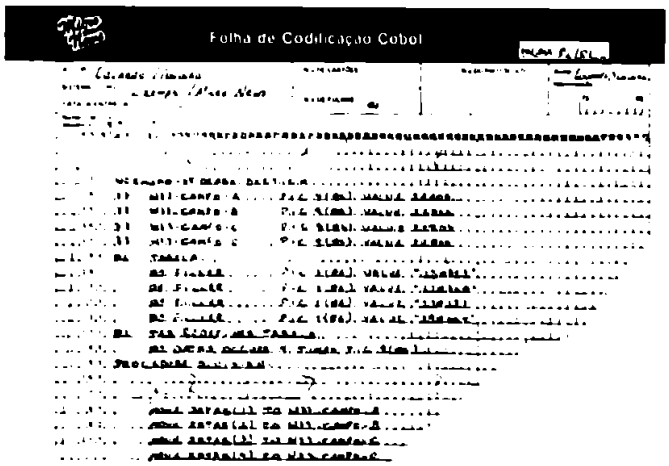
Mas ainda há um problema: se disséssemos ao COBOL que o campo MESES ou MES (que é seu único subnível) ocorre 12 vezes, como determinar cada uma de suas ocorrências? Para isso é necessário o uso de um *índice.*

Os índices podem ser de dois tipos: *indexadores* ou *subescritores.*

- Indexadores

Não é necessária sua codificação na WORKING. O COBOL se encarregará de gerá-lo. Ele permite que um índice seja indicado por uma expressão numérica. O uso de índices em seguida a um campo de tabela transforma esse campo em um campo comum.

*Exemplo:*



Nesse exemplo os índices das tabelas são literais, ou seja, movimentamos diretamente a ocorrência da tabela para onde queremos pois sabemos o que ela contém e o que fazer com esse dado.

Após a execução desse programa, temos:

W77 - CAMPO - A = 120883
W77 - CAMPO - B = 110760
W77 - CAMPO - C = 230187
W77 - CAMPO - D = 290664

Não é sempre que conhecemos qual das ocorrências da tabela vamos utilizar, por isso, temos ainda os índices de auxílio na manipulação de tabelas.

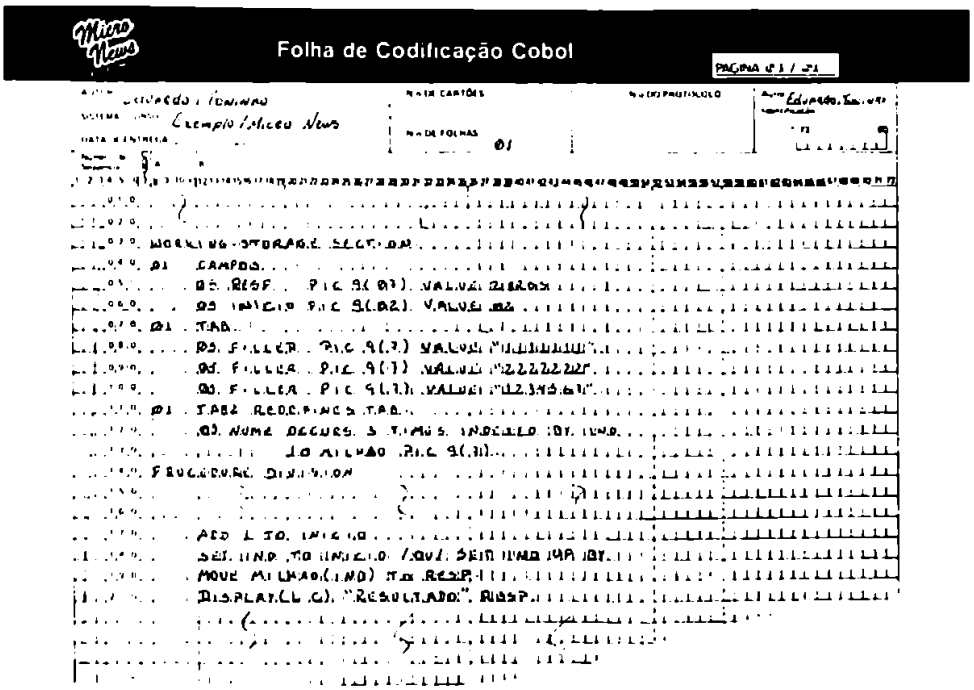
Vamos agora conhecer o comando **SET**, que na verdade é o único que pode trabalhar com indexadores. Pode ser utilizado com **MOVE, ADD** ou **SUBTRACT.**

FORMATO:

- **SET** [*literal* / *campo*] **TO** *indexador* (Para movimentação).
- **SET** *indexador* **UP BY** [*campo* / *literal*] (Para adições).
- **SET** *indexador* **DOWN BY** [*campo* / *literal*] (Para subtrações).

Um indexador não precisa ser necessariamente especificado como uma literal.

*Exemplo:* Movimentação do valor 1234567 para um campo resultado.




Após a execução IND estará valendo 3, pois é a terceira ocorrência de nossa tabela.

- Subescritores

Na cláusula **OCCURS** não constará **INDEXED BY IND**, pois o índice a ser usado estará na **WORKING-STORAGE SECTION.**

Características:

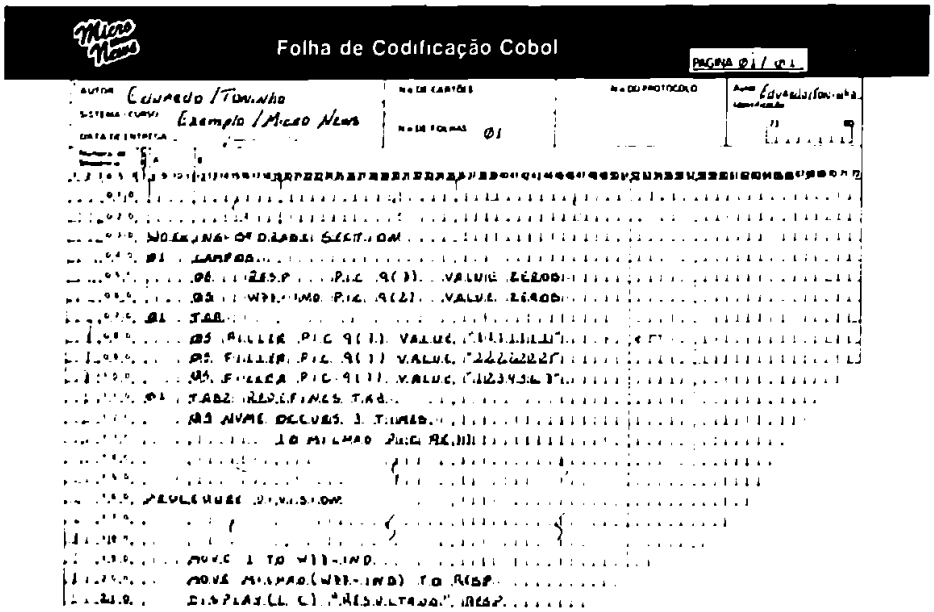
1 - É tratado no programa como um campo comum, porém obrigatoriamente numérico.

2 - Não permite índices relativos, ou seja, determinarão de seu valor através de cálculos.

3 - O comando **SET** pode ser utilizado para mudar seu conteúdo, sendo também possível quaisquer outros comandos de manipulação de campos.

4 - Ao contrário do indexador, um subescritor pode ser usado para referenciar várias tabelas, enquanto o indexador apenas aquela a que está ligado através da cláusula OCCURS INDEXED BY.

*Exemplo:*




Aparecerá o valor 1111111 de acordo com a movimentação para o subescritor (W77 - IND).

### 6.15) Comando PERFORM

FORMATO: **PERFORM** *nome - do - parágrafo.*

Ao desviarmos a sequência de instruções que o COBOL está executando para outro parágrafo, ele executará a nova sequência de instruções e ao final dessa, deverá retornar ao próximo comando após o **PERFORM.**

Podemos incrementar um conjunto de opções ao **PERFORM.**

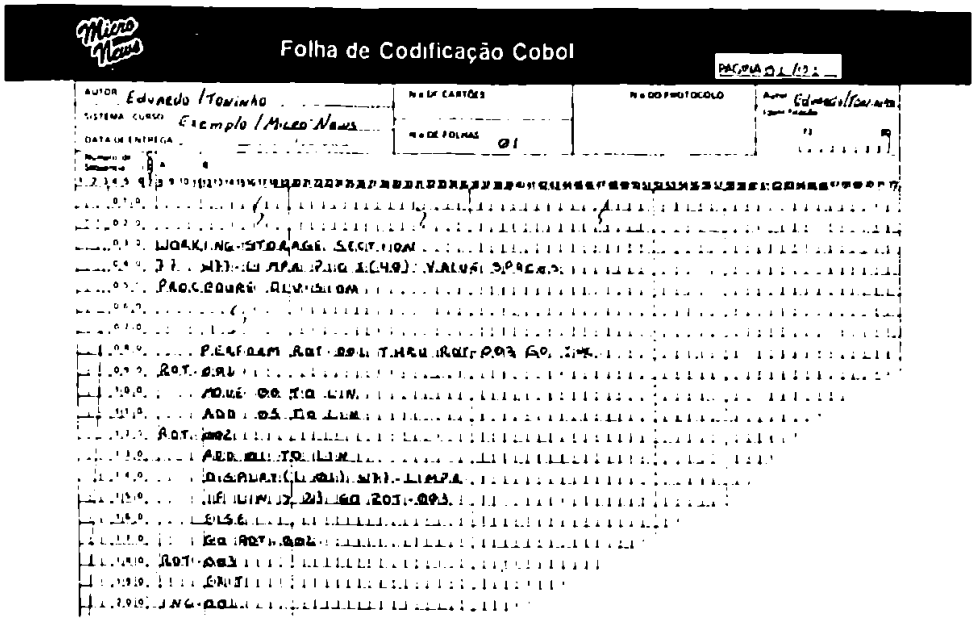
### 6.15.1) PERFORM THRU

FORMATO: **PERFORM** *nome - do - parágrafo1* **THRU** *nome - do - parágrafo2.*

Desvia para a rotina determinada em *nome - do - parágrafo1* porém não retorna ao detectar *nome - de - parágrafo2*, mas somente após sua execução.

OBS.: Os dois parágrafos deverão estar orbigatoriamente em sequência.

*Exemplo:* Limpeza de tela (sem pano e sem álcool).



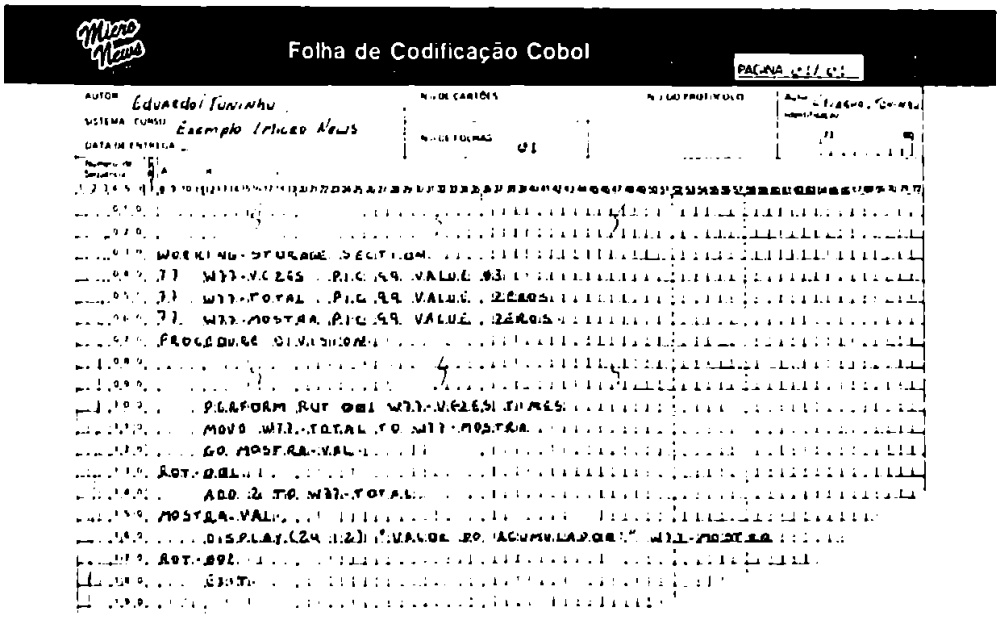

No exemplo acima, a rotina de limpeza de tela será acionada através do comando **PERFORM** da rotina ROT-ØØ1 à rotina ROT-ØØ3 até que a condição seja satisfeita.

### 6.15.2) PERFORM TIMES

FORMATO: **PERFORM** *nome-do-parágrafo campo* ou *literal numérico* **TIMES**

Desvia para a rotina determinada por nome-do-parágrafo e a executa seguidamente o número de vezes especificado.

*Exemplo:*



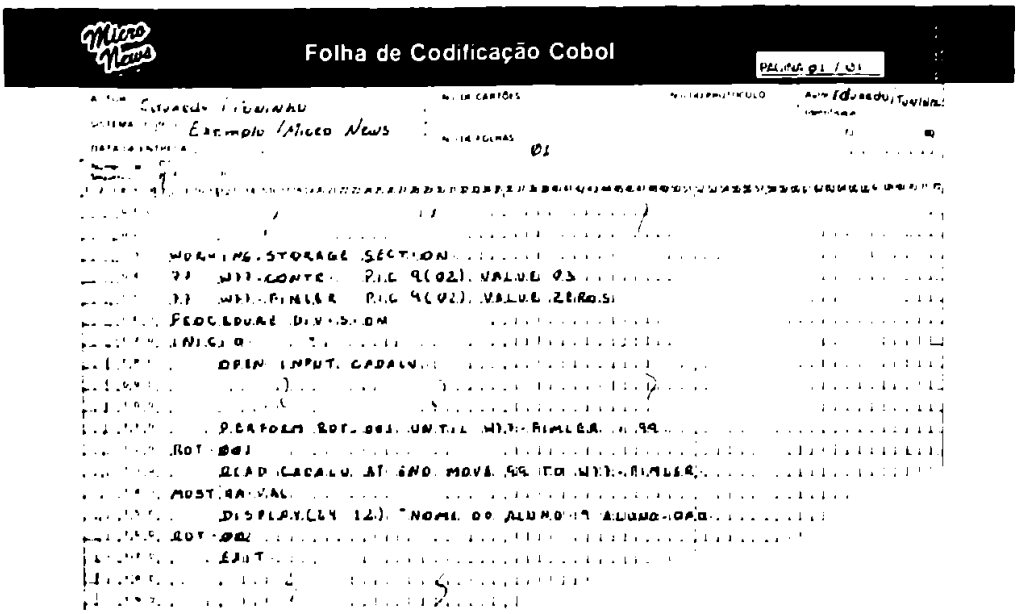
Neste exemplo, ao encontrar o **PERFORM** valerá o desvio para a rotina ROT-ØØ1 executando-a o número de vezes especificado pelo conteúdo do campo W77 - VALOR, portanto, ao encontrar a terceira execução o campo W77 - VALOR estará valendo Ø6.

### 6.15.3) PERFORM UNTIL

FORMATO: **PERFORM** *nome - do - parágrafo* **UNTIL** *condição - de - retorno*

Desvia para a rotina determinada por *nome - do - parágrafo* e a executa até que a *condição - de - retorno* torne-se verdadeira.

*Exemplo:*




Neste exemplo, ao encontrar o **PERFORM** haverá um desvio para a rotina ROT-ØØ1 executando o parágrafo de leitura até encontrar o fim do arquivo, quando o campo W77 - FIMLER for igual a 99, isto após a cobertura do arquivo conforme o parágrafo INICIO da **PROCEDURE DIVISION.**

OBS.: Caso a condição não seja satisfeita, o COBOL continuará a execução até que o operador intervenha.

### 6.15.4) PERFORM VARYING

FORMATO: **PERFORM** *nome - do - parágrafo* **VARYING** *indexador* ou *subescritor* **FROM** *literal - numérica1* **BY** *literal - numérica2* **UNTIL** *condição.*

Desvia para a rotina determinada por *nome - do - parágrafo* e a executa até que a *condição de retorno* seja obedecida, variando um *índice* (subescritor ou indexador), com o valor para a primeira execução estabelecido em *literal - numérica1* e para as demais com *índice* estabelecido por *literal - numérica2.*

*Exemplo:*

# Folha de Codificação Cobol

PAGINA 01 / 01

| | | | |
|---|---|---|---|
| AUTOR: Eduardo / Toninho | N.o DE CARTÕES: – | N.o DO PROTOCOLO | Autor: Eduardo / Toninho |
| SISTEMA / CURSO: Exemplo / Micro News | N.o DE FOLHAS: 01 | | Identificação 73 80 |
| DATA DE ENTREGA: ___ / ___ / ___ | | | |

| Número de Sequencia (1–6) | CONT (7) | A (8–11) | B (12–19) | (20–35) | (36–51) | (52–72) |
|---|---|---|---|---|---|---|
| 010 | | | | | | |
| 020 | | | | | | |
| 030 | | WORK | ING-STORAGE SECTION. | | | |
| 040 | | 77 | W77-VALIDOS | PIC X(10) | VALUE "SEGUNDA". | |
| 050 | | 01 | TAB. | | | |
| 060 | | | 05 | | | |
| 070 | | | 05 | | | |
| 080 | | | 05 | | | |
| 090 | | | 05 | | | |
| 100 | | | 05 | | | |
| 110 | | | 05 | | | |
| 120 | | | 05 | | | |
| 130 | | 01 | TABELA REDEFINES TAB INDEXED BY IND. | | | |
| 140 | | | 05 DIA OCCURS 7 TIMES PIC X(10). | | | |
| 150 | | 01 | AREA OCCURS 7 TIMES. | | | |
| 160 | | | 05 AREA-SAIDA | PIC X(10). | | |
| 170 | | PROC | EDURE DIVISION. | | | |
| 180 | | | | | | |
| 190 | | | | | | |
| 200 | | | PERFORM | PESQUISA VARYING | IND FROM 1 BY 1 | |
| 210 | | | | UNTIL W77-VALIDOS = DIA(IND). | | |
| 220 | | | | GO MOVIMENTA | | |
| 230 | | PESQ | UISA. | | | |
| 240 | | | MOVE DIA(IND) TO AREA-SAIDA(IND) | | | |
| 250 | | MOVI | MENTA. | | | |

Neste exemplo, ao encontrar o **PERFORM** haverá um desvio para a rotina PESQUISA até que o dia da tabela TAB indexada por IND seja igual ao valor do campo W77-VALIDOS. Durante a execução dessa rotina IND varia de 1 em 1. Paralelamente o programa estará preenchendo a tabela ÁREA com todas as ocorrências da tabela TAB até encontrar a condição satisfeita. Ao obter a condição, todos os dias de segunda a sexta estarão na tabela ÁREA, pois ao mover internamente o valor acumulado no índice 1, que é referente à primeira ocorrência na tabela (SEGUNDA), ou seja, quando o valor de IND for 1, W77 - VALIDOS = SEGUNDA, a condição se tornará verdadeira e a rotina não será mais executada.

### 6.15.5) Comando EXIT

Finaliza a execução de um **PERFORM**. Deve ser único dentro do parágrafo.

FORMATO: **EXIT.**

Espaço reservado para anotações de aula.

Espaço reservado para anotações de aula.

# APÊNDICE A

# AS PALAVRAS RESERVADAS DO COBOL

Segue neste apêndice um panorama de todas as palavras reservadas do COBOL.

Consulte-o sempre que estiver em dúvida se determinada palavra escolhida por você pode ou não ser usada como palavra do programador.

Para aqueles que possuem certa dificuldade de memorização, seria um bom exercício anotar em frente a cada palavra seu significado e em alguns casos apenas sua tradução para o português. Isto facilitaria uma futura consulta. Para tanto, reservamos em frente a cada palavra um espaço pára ser utilizado por você aluno neste trabalho.

Divirta-se!

A

ACCEPT _____

ACCESS _____

ADD _____

ADVANCING _____

AFTER _____

ALL _____

ALPHABETIC _____

ALTER _____

AND _____

ARE _____

AREA _____

AREAS _____

ASCENDING _____

ASCII _____

ASSIGN _____

AT _____

AUTHOR _____

AUTO _____

AUTO-SKIP _____

B

BACKGROUND-COLOR _____

BEEP _____

BEFORE _____

BELL _____

BLANK _____

BLINK _____

BLOCK _____

BOTTOM _____

BY _____

C

CALL _____

CHAIN _____

CHAINING _____

CHARACTER _____

CHARACTERS _____

CLOSE _____

CODE-SET _____

COL _____

COLLATING _____

COLUMN _____

COMMA _____

COMP _____

COMPUTATIONAL _____

COMP-0 _____

COMPUTATIONAL-0 _____

COMP-3 _____

COMPUTATIONAL-3 _____
COMPUTE _____
CONFIGURATION _____
CONTAINS _____
COPY _____
COUNT _____
CURRENCY _____

**D**

DATA _____
DATE _____
DATE-COMPILED _____
DATE-WRITTEN _____
DAY _____
DEBUGGING _____
DECIMAL-POINT _____
DECLARATIVES _____
DELETE _____
DELIMITED _____
DELIMITER _____
DEPENDING _____
DESCENDING _____
DISK _____
DISPLAY _____
DIVIDE _____
DIVISION _____
DOWN _____
DYNAMIC _____

**E**

ELSE _____
EMPTY-CHECK _____
END _____
END-OF-PAGE _____
ENVIRONMENT _____
EOP _____
EQUAL _____
ERASE _____
ERROR _____
ESCAPE _____
EXCEPTION _____
EXHIBIT _____
EXIT _____
EXTEND _____

**F**

FD _____
FILE _____
FILE-CONTROL _____
FILE-ID _____
FILLER _____
FIRST _____
FOOTING _____
FOR _____
FOREGROUND-COLOR _____
FROM _____
FULL _____

**G**

GIVING _____
GO _____
GREATER _____

H

HIGHLIGHT

HIGH-VALUE

HIGH-VALUES

I

IDENTIFICATION

IF

IN

INDEX

INDEXED

INITIAL

INPUT

INPUT-OUTPUT

INSPECT

INSTALLATION

INTO

INVALID

IS

I-O

I-O-CONTROL

J

JUST

JUSTFIED

K

KEY

L

LABEL

LEADING

LEFT

LEFT-JUSTIFY

LENGTH-CHECK

LESS

LIN

LINAGE

LINAGE-COUNTER

LINE

LINES

LINKAGE

LOCK

LOW-VALUE

LOW-VALUES

M

MEMORY

MERGE

MODE

MODULES

MOVE

MULTIPLY

N

NAMES

NATIVE

NEGATIVE

NEXT

NO-ECHO

NOT

NUMBER

NUMERIC

O

OBJECT-COMPUTER
OCCURS
OF
OMITTED
ON
OPEN
OR
ORGANIZATION
OUTPUT
OVERFLOW

P

PAGE
PERFORM
PIC
PICTURE
PLUS
POINTER
POSITIVE
PRINTER
PROCEDURE
PROCEDURES
PROCEED
PROGRAM
PROGRAM-ID
PROMPT

Q

QUOTE
QUOTES

R

RANDOM
READ
READY
RECORD
RECORDS
REDEFINES
RELATIVE
RELEASE
REMOVAL
REPLACING
REQUIRED
RERUN
RESERVE
RESET
RETURN
REVERSE-VIDEO
REWRITE
RIGHT
RIGHT-JUSTIFY
ROUNDED
RUN

S

SAME
SCREEN
SEARCH
SECTION

SECURE
SECURITY
SELECT
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGE
SOURCE-COMPUTER
SPACE
SPACES
SPACE-FILL
SPECIAL-NAMES
STANDARD
STANDARD-1
START
STATUS
STOP
STRING
SUBTRACT
SWITCH-1
SWITCH-2
SWITCH-3
SWITCH-4
SWITCH-5
SWITCH-6
SWITCH-7
SWITCH-8
SYNC
SYNCHRONIZED

T

TALLYING
THEN
THROUGH
THRU
TIME
TIMES
TO
TOP
TRACE
TRAILING
TRAILING-SIGN

U

UNDERLINE
UNSTRING
UNTIL
UP
UPDATE
UPON
USAGE
USE
USER
USING

V

VALUE

VALUES _____

VARYING _____

**W**

WHEN _____

WITH _____

WORDS _____

WORKING-STORAGE _____

WRITE _____

**Z**

ZERO _____

ZEROES _____

ZEROS _____

ZERO-FILL _____

Espaço reservado para anotações de aula.

# APÊNDICE B
# EXERCÍCIOS

1 - Explique a estrutura de um arquivo. Dê um exemplo.

2 - Qual a diferença entre um registro físico e um registro lógico? Quando o registro físico coincide com o registro lógico?

3 - Descreva (tipo e tamanho) dos seguintes campos:

- Sigla de Estado: _____
- Idade de Funcionário: _____
- Nome de Aluno: _____
- RG: _____
- CIC: _____

4 - O que significa COBOL e qual sua aplicação no mercado?

5 - Qual a estrutura de um programa COBOL? Por que o COBOL possue divisões internas enquanto o BASIC não?

6 - Qual a função das constantes figurativas? Dê dois exemplos.

7 - Qual a função da folha de codificação? Por que em COBOL ela é usada e em BASIC não?

8 - Dê um exemplo de IDENTIFICATION DIVISION usando somente as especificações obrigatórias.

9 - Qual a função do SPECIAL-NAME DECIMAL-POINT IS COMMA?

10 - Explique a cláusula SELECT.

11 - Qual a função dos números e indicadores de nível?

12 - Descreva a cláusula PICTURE.

13 - Monte a FD do seguinte arquivo:

- Nome interno: ARQ-AMIGO.
- Gravado em disco com o nome de AMIGOS. TXT.
- Registro: REG-AMI. Fator de bloco = 5.
- Campos:

| | | |
|---|---|---|
| NOME | ALFANUMÉRICO | 40POSIÇÕES |
| ENDEREÇO | ALFANUMÉRICO | 40 POSIÇÕES |
| TELEFONE | NUMÉRICO | 07 POSIÇÕES |
| DATA-NASC | ALFANUMÉRICO | 08 POSIÇÕES |

A | B

14 - Codifique as três primeiras divisões de um programa seguindo as instruções abaixo:

- Arquivo de Entrada: CAD-FUNC.

  Gravado em disco com o nome CAD88.TXT.

  Registro: REG-FUNC

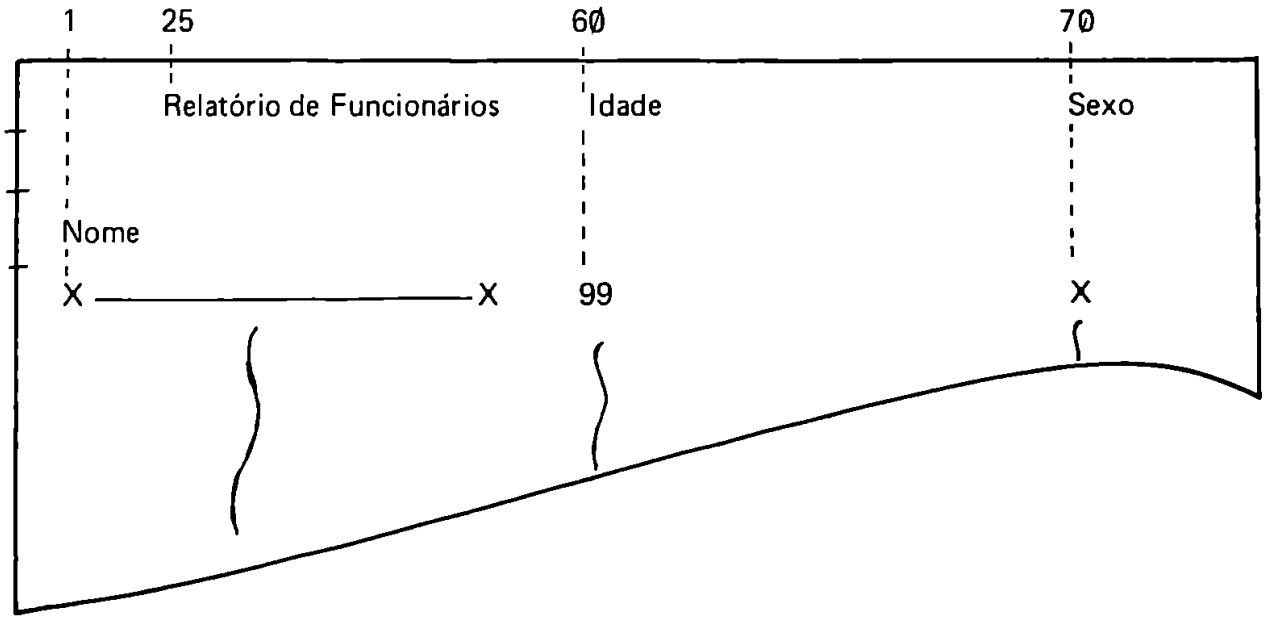
  Campos: NOME X(4Ø).

  IDADE 9(Ø2).

  SEXO X(Ø1).

- Arquivo de Saída: RELATÓRIO.

  Saída em impressora conforme LAY-OUT:




- Campo auxiliar: CONT-LIN 9(Ø3).

15 - O que é um arquivo sequencial e qual sua utilização?

16 - O que é um arquivo de organização indexada?

17 - Explique organização e acesso. Dê exemplos.

18 - Diga para que servem os seguintes comandos. Dê os formatos e exemplos.

- PERFORM:
- FILE STATUS:
- WRITE:
- REWRITE:
- DELETE:

- READ: _____

_____

- START: _____

_____

- OPEN: _____

_____

- BLANK SCREEN: _____

_____

19 - Descreva a diferença entre os seguintes comandos:

- OPEN OUTPUT *arquivo* / OPEN I-O *arquivo*

_____

_____

_____

- READ *arquivo* AT END / READ *arquivo* NEXT RECORD AT END

_____

_____

_____

2Ø - Descreva a SELECT para:

- Arquivos de organização sequencial.

_____

_____

_____

_____

_____

- Arquivos de organização indexada.

_____

_____

_____

_____

21 - Elabore um programa (com respectivos arquivos) que permita as seguintes operações sobre um arquivo de clientes de um banco.

a - Abertura de conta.

b - Consulta de saldo.

c - Depósito.

d - Retirada.

Espaço reservado para anotações de aula.

Espaço reservado para anotações de aula.

# APÊNDICE C
# DESENVOLVIMENTO DE UM SISTEMA

Desenvolva um programa seguindo o roteiro abaixo:

1 - Arquivo de organização indexada pelo código do produto e manutenção dos dados contendo Inclusão/ Alteração/ Exclusão/ Consulta (IAEC).

2 - Seguir a orientação dos LAY-OUTs conforme especificado, pois esse programa servirá de base para um sistema de controle de estoque onde numa segunda etapa faremos a parte de relatórios.

- Inclusão – 1.º grupo.
- Alteração – 2.º grupo.
- Exclusão – 3.º grupo.
- Consulta – 4.º grupo.

3 - Lay-outs do registro.

| CÓDIGO DO PRODUTO | NOME DO PRODUTO | VALOR UNIT. | EST. MINIMO |
|---|---|---|---|
| 99999 | X(2Ø) | 99999,99 | 9999,99 |
| REFERÊNCIA | DATA DE ENTREGA | NOME DO FORN. | PESO UNIT. |
| X(Ø9) | 99/99/99 | X(3Ø) | 999,99 |

4 - Procedimentos para o programa de manutenção.

4.1 - O sistema deverá possuir uma tela de menu principal para escolha de funções.
4.2 - O sistema terá também telas para inclusão, alteração, exclusão e consulta.
4.3 - A tela deverá ser fixa e só se apagará após a aceitação via teclado dos dados nos campos respectivos.
4.4 - Lay-out do seletor principal.

```
DATA: 99/99/99      MICRO NEWS – CURSO DE COMPUTAÇÃO      VERSÃO 1.Ø

    * SISTEMA DE CONTROLE DE ESTOQUES DE PRODUTOS – I/A/E/C *
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      ( Ø1 )  –  Inclusão no Cadastro de Produtos
      ( Ø2 )  –  Alteração no Cadastro de Produtos
      ( Ø3 )  –  Exclusão no Cadastro de Produtos
      ( Ø4 )  –  Consulta no Cadastro de Produtos
      ( Ø5 )  –  Emissão de Relatórios Totais de Estoques
      ( Ø6 )  –  Emissão de Relatórios Totais em Baixa no Estoque
      ( =>)   –  Digite a opção . . . : (     )




MENSAGEM:
```

4.5 - O campo de opção deverá possuir apenas uma posição numérica; não poderá ser menor do que 1 ou maior do que 6. Caso isso aconteça, deverá ser exibida a mensagem OPÇÃO INVALIDA TECLE <ENTER> devendo retornar e aceitar nova opção.

```
DATA: 99/99/99      MICRO NEWS CURSO DE COMPUTAÇÃO      VERSÃO 1.Ø

** SISTEMA DE CONTROLE DE ESTOQUES DE PRODUTOS – INC/ALT/EXC/CON
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

      ( ØØ ) –  Código do Produto . . . . . .:
      ( Ø1 ) –  Nome do Produto . . . . . .:
      ( Ø2 ) –  Valor Unitário . . . . . .:
      ( Ø3 ) –  Peso Unitário (kg) . . . . . .:
      ( Ø4 ) –  Estoque Mínimo . . . . . .:
      ( Ø5 ) –  Quantidade Em Estoque . . . . . .:
      ( Ø6 ) –  Referência do Produto . . . . . .:
      ( Ø7 ) –  Data de Entrada No Estoque . . . . . .:
      ( Ø8 ) –  Nome do Fornecedor . . . . . .:

      MENSAGEM:
```

4.6 - Toda vez que o operador terminar de cadastrar, consultar, excluir ou alterar e quiser retornar ao seletor deverá teclar < ENTER > no código do produto e automaticamente deverá voltar ao MENU principal.

4.7 - Quando código do produto for igual a zero deverá voltar para a tela principal do sistema.

4.8 - Nome do Produto deverá ser diferente de espaços.

4.9 - Valor unitário deverá ser diferente de zero.

4.1Ø - Estoque mínimo deverá ser diferente de zero.

4.11 - Referência do produto deverá ser diferente de espaços.

4.12 - Data de entrega no estoque deverá ser a data de compra do produto. Obrigatoriamente diferente de zero.

4.13 - Nome do fornecedor deverá ser diferente de espaços.

4.14 - O campo mensagem deverá trocar de valor a cada passo do processamento.

4.15 - Mensagens

4.15.1 - Inclusão:

- Produto já cadastrado tecle < ENTER > .
- Caso a mensagem venha a ser erro de gravação deverá apresentar o FILE STATUS correspondente, fechar os arquivos e interromper a execução.

4.15.2 - Alteração:

- Produto não cadastrado tecle < ENTER > .
- Erro de gravação no arquivo *arquivo*. Deverá apresentar o FILE STATUS correspondente, fechar os arquivos e interromper a execução.

4.15.3 - Exclusão:

- Produto não cadastrado tecle<ENTER. >
- Erro de deleção no arquivo *arquivo.* Deverá apresentar o FILE STATUS correspondente, fechar os arquivos e interromper a execução do programa.

4.15.4 - Consulta:

- Produto não cadastrado tecle<ENTER. > .
- Após a consulta do ítem deve perguntar se deseja outra consulta. Em caso de resposta negativa deverá voltar para a tela principal.

Em todos os casos, o retorno deverá ser feito para a tela principal quando o operador não desejar mais continuar a operação.

5 - Lay-out de relatório.

| | | |
|---|---|---|
| DATA: 99/99/99 | MICRO NEWS CURSO DE COMPUTAÇÃO | VERSÃO 1.Ø |
| SISTEMA DE CONTROLE DE ESTOQUES DE PRODUTOS | | – PÁGINA: 999 |

5.1 - Todos os relatórios deverão seguir lay-out acima.

5.2 - Todas as páginas deverão conter cabeçalhos.

5.3 - Relatórios sobre totais de estoque deverão conter a somatória em valor monetário, do estoque, isto é, deverá fornecer uma posição analítica do valor em dinheiro e do valor em peso.

5.4 - A partir da quantidade mínima em estoque deverá ser emitido no final do mês um relatório evidenciando quais os produtos que se encontram em baixa no estoque.

Boa sorte!

Espaço reservado para anotações de aula.

Espaço reservado para anotações de aula.

COBOL

# Micro News

CURSO DE COMPUTAÇÃO

O ELO ENTRE O HOMEM E A MÁQUINA